Dynamic Coded Caching in Wireless Networks

Dynamic Coded Caching in Wireless Networks

Jesper Pedersen, Alexandre Graell i Amat, *Senior Member, IEEE*, Jasper Goseling, *Member, IEEE*, Fredrik Brännström, *Member, IEEE*, Iryna Andriyanova, *Member, IEEE*, and Eirik Rosnes, *Senior Member, IEEE*

***Abstract*—We consider distributed and dynamic caching of coded content at small base stations (SBSs) in an area served by a macro base station (MBS). Specifically, content is encoded using a maximum distance separable code and cached according to a time-to-live (TTL) cache eviction policy, which allows coded packets to be removed from the caches at periodic times. Mobile users requesting a particular content download coded packets from SBSs within communication range. If additional packets are required to decode the file, these are downloaded from the MBS. We formulate an optimization problem that is efficiently solved numerically, providing TTL caching policies minimizing the overall network load. We demonstrate that distributed coded caching using TTL caching policies can offer significant reductions in terms of network load when request arrivals are bursty. We show how the distributed coded caching problem utilizing TTL caching policies can be analyzed as a specific single cache, convex optimization problem. Our problem encompasses static caching and the single cache as special cases. We prove that, interestingly, static caching is optimal under a Poisson request process, and that for a single cache the optimization problem has a surprisingly simple solution.**

***Index Terms*—Caching, content delivery networks, erasure correcting codes, TTL.**

I. Introduction

Caching has attracted a significant amount of attention in the last few years as a promising technology to alleviate the load on backhaul links [1]. Content may be cached in a distributed fashion across small base stations (SBSs) such that users can download requested content directly from them. For distributed caching, the use of erasure correcting codes (ECCs) has been shown to reduce the download delay as well as the network load [2], [3]. Content may also be cached directly in mobile devices such that users can download content from neighboring devices using device-to-device (D2D) communication. Similar to the SBS caching case, the use of ECCs has been demonstrated to reduce the network load also for this scenario [4]–[6]. Caching furthermore facilitates index-coded broadcasts to multiple users requesting different content, which has been shown to drastically reduce the amount of data that has to be transmitted over the SBS-to-device downlink [7]. ECCs have also been used in conjunction with index coding to ensure data availability at broadcasting devices in D2D caching networks [8] as well as to provide index-coding broadcast opportunities to subsets of users [9], [10]. All these works consider the cached content to be static for a period of time (e.g., a day) according to a given file popularity distribution.

Dynamic cache eviction policies, e.g., first-in-first-out (FIFO), least-recently-used (LRU), least-frequently-used (LFU), and random (RND), may be beneficial to use when the file library or file popularity profile is dynamic, or when users request content according to a renewal process [11]. Due to the complexity in analyzing such policies, timer-based policies that are significantly more tractable have been suggested. One such policy is time-to-live (TTL) where a request for a particular piece of content triggers it to be cached and then evicted after the expiration of a timer. The TTL policy has been shown to yield similar performance to FIFO, LRU, LFU, and RND policies in [12]–[15]. Goseling and Simeone extended the TTL policy to cache fractions of files, referred to as fractional TTL (FTTL), and showed that this can improve performance under a renewal request process [16]. Decreasing the fraction of a file that is cached over time, termed soft TTL (STTL), can further improve the performance. Optimal STTL caching policies are obtained through a convex optimization problem [16]. All previous works on TTL policies assume either a single cache or a number of caches, e.g., structured into lines or hierarchies, where users access a single cache. For these scenarios, coded caching does not bring any benefits. However, if users can access several caches, the use of ECCs can be beneficial. Hence, merging distributed coded caching with the TTL schemes in [16], which have both independently been shown to bring performance improvements, is an intriguing prospect.

In this paper, we generalize the TTL policies in [16] to a distributed coded caching scenario. Specifically, we consider the scenario where content is encoded using a maximum distance separable (MDS) code and cached in a distributed fashion across several SBSs. Coded content is evicted from the caches in accordance with the TTL policies in [16]. Users requesting a particular piece of content download coded packets from SBSs within communication range and, if necessary, download additional packets from a macro base station (MBS). The main contributions are summarized below.

A. Contributions

We generalize the TTL, FTTL, and STTL caching policies in [16] to a scenario where coded packets are cached in a distributed fashion across several SBSs. Specifically, we adopt the maximization problem in [16], providing optimal caching

This work was funded by the Swedish Research Council under grant 2016-04253 and by the National Center for Scientific Research in France under grant CNRS-PICS-2016-DISCO.

J. Pedersen, A. Graell i Amat, and F. Brännström are with the Department of Electrical Engineering, Chalmers University of Technology, SE-41296 Gothenburg, Sweden (e-mail: {jesper.pedersen, alexandre.graell, fredrik.brannstrom}@chalmers.se).

J. Goseling is with the Department of Applied Mathematics, University of Twente, 7522 Enschede, The Netherlands (e-mail: j.goseling@utwente.nl).

I. Andriyanova is with the ETIS-UMR8051 group, CY Cergy Paris University/ENSEA/CNRS, France (e-mail: iryna.andriyanova@ensea.fr).

E. Rosnes is with Simula UiB, N-5020 Bergen, Norway (e-mail: eirikrosnes@simula.no).

policies for a single cache and an increasing and concave utility function. We modify the objective function of the problem in [16] to yield a network load minimization problem, where the network load is defined as a sum of data rates over various network links, weighted by a cost representing, e.g., transmission delay or energy consumption of transmitting data over these links. We then rewrite the optimization problem as a mixed integer linear program (MILP) that is efficiently solved numerically. We furthermore prove that the distributed coded caching problem can equivalently be analyzed as a single cache problem with a specific decreasing and convex cost function. This is an important result because it shows that such a function, previously studied for the single cache case due to its analytical tractability [16], arises naturally in a distributed caching scenario. For SBSs deployed according to a Poisson point process [17, Ch. 2.3], we derive the cost function explicitly. We analyze two important special cases of the network load minimization problem. In particular, we show that our problem has the static coded caching problem where content is never updated (considered in, e.g., [2], [3]), as a special case. We furthermore prove that static coded caching is optimal under the assumption of a Poisson request process. Moreover, for the special case of users accessing a single cache, we prove that the STTL problem is a fractional knapsack problem with a greedy optimal solution. The performance of TTL, FTTL, and STTL, in terms of network load, is evaluated for a renewal process, specifically when the times between requests follow a Weibull distribution, which has been shown to accurately model requests for, e.g., educational media [18]. We show that distributed coded caching using TTL caching policies can offer significant reductions in network load, especially for bursty renewal request processes.

B. Related Work

Distributed caching of coded content utilizing TTL cache eviction policies was also investigated in [19]. Compared to the problem studied in this paper, the work in [19] is significantly different in a number of ways. Specifically, we consider an STTL policy with optimized TTL timers under a renewal request process, which was not considered in [19]. Furthermore, a dynamic library of files with location-dependent popularity is considered in [19], which is typically considered to be more general than a static file library and is not in the scope of our work. However, it is reasonable to consider scenarios where the file library remains fixed for a considerable amount of time, e.g., a day, and focus on an area with homogeneous file popularity.

II. System Model

We consider an area served by an MBS that always has access to a file library of N files, where file $i = 1, 2, \ldots, N$ has size s_i. Mobile users request files from the library according to independent renewal processes. Specifically, we denote the independent and identically distributed times between requests for file i by X_i, the cumulative distribution function (CDF) of X_i by

$$F_{X_i}(t) \triangleq \Pr(X_i \leq t),$$

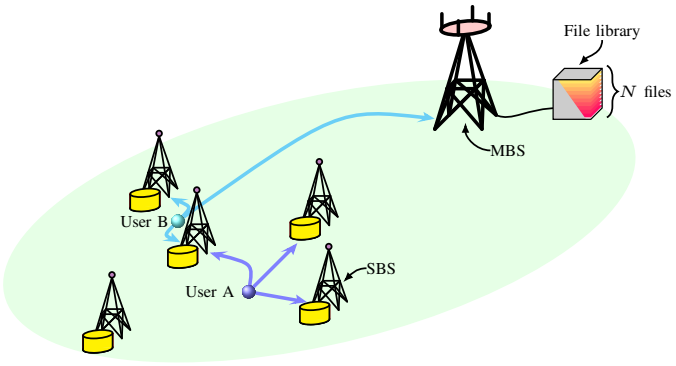

Figure 1. The considered network with an MBS with access to a file library of N files, a number of SBSs caching coded fractions of files, and users A and B. In the example, user A can decode the requested file by downloading coded packets from the SBSs within communication range, whereas user B has to also download some packets from the MBS.

and the request rate of file i by

$$\omega_i \triangleq \mathbb{E}[X_i]^{-1}.$$

We let $p_i = \omega_i/\omega$, for some aggregate request rate in the area, $\omega = \sum_{i=1}^{N} \omega_i$. For a Poisson request process, i.e., exponentially distributed X_i, p_i can be interpreted as the probability that file i is requested. The request rates ω_i are assumed to be constant over a sufficiently long period of time, e.g., not changing during the course of one day. For such scenarios, file popularity predictions and content allocation optimization can be carried out during periods of low network traffic, e.g., during night time. B SBSs are deployed in the area and each SBS has a cache with storage capacity C. We assume that a user can download content from an SBS if it is within a range r_{SBS} and we denote by γ_b the probability that a user is within range of b SBSs at any given time. The model considered in this paper is illustrated in Fig. 1.

A. Caching Policy

Each file i of size s_i is partitioned into k_i packets, each of size s_i/k_i. The packets are encoded into n_i coded packets (also of size s_i/k_i) using an (n_i, k_i) MDS code of rate k_i/n_i. We defer the description of how to select code parameters n_i and k_i until later in this subsection. For analytical tractability, we assume that all SBSs cache the same amount of each file at all times, i.e., the same number of distinct coded packets for a given file. In this respect, the caches are *synchronized*. With slight abuse of notation, we let $m_i(t) \leq k_i$ denote the number of distinct coded packets of file i cached at each SBS at time t, where t is the time since the last request for file i. We will use this interpretation of t throughout the paper. The amount of file i cached by each SBS at time t is hence $m_i(t)s_i/k_i$. We normalize by the file size s_i and let

$$\mu_i(t) = m_i(t)/k_i$$

denote the fraction of file i cached at time t. Similar to [16], we refer to $\mu_i(t)$ as the *caching policy*.

We adopt an STTL cache eviction policy, shown to increase the amount of content that can be downloaded from a single cache under a renewal request process in [16]. Hence, coded

packets of file i may be evicted from the caches at periodic times with period T after the last request for file i. We allow K potential updates within a total time equal to KT, which we refer to as the *update window* length. For $K = 0$, the caches are never updated. This corresponds to *static caching*, which is the type of caching considered in a big part of the literature [2]–[7]. The caching policies $\mu_i(t)$ are decreasing functions of t given by [16]

$$\mu_i(t) = \begin{cases} \mu_{i,0}, & \text{if } t < T, \\ \mu_{i,j}, & \text{if } jT \leq t < (j+1)T,\ j = 1, 2, \ldots, K-1, \\ \mu_{i,K}, & \text{if } t \geq KT, \end{cases} \tag{1}$$

where

$$1 \geq \mu_{i,0} \geq \mu_{i,1} \geq \cdots \geq \mu_{i,K} \geq 0.$$

To derive code parameters k_i and n_i from caching policy $\mu_{i,j}$ for any i, we first quantize $\mu_{i,j}^{-1}$, for all j, to rational numbers to the desired degree of precision. We then obtain k_i as the least common multiple of $\mu_{i,j}^{-1}$, i.e., the smallest integer multiple of $\mu_{i,j}^{-1}$, for all j such that $\mu_{i,j} > 0$, and $n_i = Bk_i\mu_{i,0}$. After a request for file i, the n_i coded packets are allocated to the caches, where each SBS caches $k_i\mu_{i,0}$ distinct coded packets. Subsequently, coded packets are evicted from the caches in accordance with the caching policy in (1). The following small example illustrates how to obtain code parameters k_i and n_i from an STTL caching policy and how to manage cached content over time.

Example 1. *Consider a small network with three SBSs ($B = 3$) and the STTL caching policy*

$$\mu_{i,j} = \begin{cases} 1, & \text{if } j = 0, \\ 2/3, & \text{if } j = 1, 2, \ldots, 4, \\ 1/3, & \text{if } j = 5, \\ 0, & \text{if } j = K = 6, \end{cases}$$

for some file i. The least common multiple of 1^{-1}, $(2/3)^{-1}$, and $(1/3)^{-1}$ is 3 and, hence, $k_i = 3$. It follows that $n_i = Bk_i\mu_{i,0} = 9$. After a request for file i, each SBS caches $k_i\mu_{i,0} = 3$ distinct coded packets of file i. If there is no request for file i until time $t = T$, one coded packet is evicted from the cache of each SBS, i.e., $k_i\mu_{i,1} = 2$ coded packets are cached at each SBS, etc..

We remark that, for MDS codes, the required underlying field size, as well as the encoding and decoding complexity, grow with n_i. If n_i is large, rateless codes with significantly lower complexity can be used [20]. Such codes have been shown to have close to MDS code performance in distributed caching scenarios [21]. In this work, we will assume that $\mu_{i,j} \in \mathbb{R}$ for simplicity. We refer to $f = 1/T$ as the update frequency and remark that static caching ($K = 0$) corresponds to $f = 0$.

B. Content Download

For an MDS code, any k_i coded packets of file i suffice to decode the file. All files requested by users can be decoded by downloading packets available at SBSs within communication range and, if necessary, retrieving additional packets from the MBS. Specifically, a user requesting file i at time t downloads $m_i(t)$ packets from the b SBSs within communication range. If $bm_i(t) \geq k_i$, the user can decode the file. If $bm_i(t) < k_i$, the additional $k_i - bm_i(t)$ coded packets required to decode the file are downloaded from the MBS. Consequently, the fraction of file i downloaded from SBSs can be expressed as

$$\min\{1, b\mu_i(t)\} \tag{2}$$

and the fraction of file i downloaded from the MBS as

$$\max\{0, 1 - b\mu_i(t)\}. \tag{3}$$

We assume that downloading one bit of data from the MBS and the SBSs comes at a cost θ_{MBS} and θ_{SBS} per bit, respectively. The cost represents, e.g., the transmission delay or energy consumption of transmitting one bit. Finally, the cost to send data to the caches, referred to as the cache update cost, is denoted by θ_{C}.

III. Preliminaries

The caching policies in (1) correspond to STTL [16]. FTTL policies are obtained as a special case of (1), where the same fraction ν_i of file i is cached for a time LT, defined by an integer $0 \leq L \leq K$, i.e., $\mu_{i,0} = \mu_{i,1} = \ldots = \mu_{i,L} = \nu_i$ and $\mu_{i,L+1} = \mu_{i,L+2} = \ldots = \mu_{i,K} = 0$ [16]. Furthermore, letting $\nu_i = 1$ we obtain TTL caching policies. In [16], the caching problem is framed as a utility maximization problem where a strictly concave and increasing utility function $g_i(\mu)$ measures the utility resulting from caching a fraction μ when file i is requested. The choice of letting $g_i(\mu)$ be a strictly concave and increasing function of μ is due to analytical tractability. In practice, a linear utility function is more reasonable [22].

For the case of a single cache, the sum utility maximization solved in [16] is

$$\underset{\substack{\mu_{i,j},\nu_i \in \mathbb{R} \\ \beta_{i,j} \in \{0,1\}}}{\text{maximize}} \quad \sum_{i=1}^{N} \omega_i \sum_{j=0}^{K} g_i(\mu_{i,j}) F_{i,j}, \tag{4}$$

$$\text{subject to} \quad \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \mu_{i,j} A_{i,j} \leq C, \tag{5}$$

$$1 \geq \mu_{i,0} \geq \mu_{i,1} \geq \cdots \geq \mu_{i,K} \geq 0, \tag{6}$$

$$0 \leq \nu_i \leq 1, \tag{7}$$

$$-\beta_{i,j} \leq \mu_{i,j} \leq \beta_{i,j}, \tag{8}$$

$$\beta_{i,j} - 1 \leq \mu_{i,j} - \nu_i \leq 1 - \beta_{i,j}, \tag{9}$$

where (5) is a long-term average cache capacity constraint [16, Lem. 1],

$$F_{i,j} = \begin{cases} F_{X_i}((j+1)T) - F_{X_i}(jT), & \text{if } j = 0, \ldots, K-1, \\ 1 - F_{X_i}(KT), & \text{if } j = K \end{cases} \tag{10}$$

is the probability that file i is requested in time-slot j, and

$$A_{i,j} = \begin{cases} \displaystyle\int_{jT}^{(j+1)T} 1 - F_{X_i}(t)\ \mathrm{d}t, & \text{if } j = 0, \ldots, K-1, \\ \displaystyle\int_{KT}^{\infty} 1 - F_{X_i}(t)\ \mathrm{d}t, & \text{if } j = K. \end{cases} \tag{11}$$

The ratio $F_{i,j}/A_{i,j} \approx h_i(jT)$, where $h_i(\cdot)$ is the hazard function of the request process, represents the probability to observe a request given the time since the last request, and the approximation follows by considering the continuous limit $T \to 0$ [16]. For the remainder of this paper, $h_i(jT)$ is assumed to be decreasing in j. The solution to (4)–(9) provides optimal FTTL caching policies [16]. Optimal TTL caching policies are achieved by letting $\nu_i = 1$ and removing the constraint (7), while STTL policies are achieved by removing the constraints (7)–(9) [16].

IV. Distributed Coded TTL Caching

In this section, we formulate the average rate at which data is sent through the network described in Section II and an optimization problem to minimize the network load for coded TTL caching. In particular, we generalize the optimization problem (4)–(9) to a distributed coded caching scenario, utilizing TTL caching policies. We propose an equivalent, more tractable formulation of the optimization problem that is efficiently solved numerically. The average rate at which data is downloaded from the SBSs and the MBS is denoted by R_{SBS} and R_{MBS}, respectively. We choose the utility function $g_i(\mu_i(t)) = s_i \min\{1, b\mu_i(t)\}$, representing the amount of data that a user requesting file i at time t can download from the b SBSs within communication range (see (2)). Using (1) and also averaging over the number of SBSs within range of a user requesting a particular content in (4), we obtain

$$R_{\text{SBS}} = \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \min\{1, b\mu_{i,j}\} F_{i,j}. \tag{12}$$

Similarly, substituting (1) in (3), the MBS download rate is

$$R_{\text{MBS}} = \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \max\{0, 1 - b\mu_{i,j}\} F_{i,j}. \tag{13}$$

Note that $\max\{0, 1 - b\mu_{i,j}\} = 1 - \min\{1, b\mu_{i,j}\}$ and that, using (10) together with $F_{X_i}(0) = 0$,

$$\sum_{j=0}^{K} F_{i,j} = F_{X_i}(KT) - F_{X_i}(0) + 1 - F_{X_i}(KT) = 1. \tag{14}$$

Hence, we may rewrite (13) as

$$\begin{aligned} R_{\text{MBS}} &= \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} (1 - \min\{1, b\mu_{i,j}\}) F_{i,j} \\ &= \sum_{i=1}^{N} \omega_i s_i - R_{\text{SBS}}, \end{aligned} \tag{15}$$

i.e., all requested content not downloaded from SBSs is downloaded from the MBS. The average data rate at which the SBSs caches are updated, denoted by R_{C}, is

$$R_{\text{C}} = B \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} (\mu_{i,0} - \mu_{i,j}) F_{i,j}.$$

The above expression assumes that all caches are updated at each request in the area, which we refer to as *synchronous* updates. This simplification is due to analytical tractability. Obtaining optimal caching policies under *asynchronous* updates appears to be a formidable task. In Section VI, we nonetheless simulate caching policies that are optimal under synchronous updates for an asynchronous cache updating scenario.

We define the *network load* as

$$W = \theta_{\text{MBS}} R_{\text{MBS}} + \theta_{\text{SBS}} R_{\text{SBS}} + \theta_{\text{C}} R_{\text{C}}, \tag{16}$$

where

$$\theta_{\text{MBS}} R_{\text{MBS}} + \theta_{\text{SBS}} R_{\text{SBS}} = \theta_{\text{MBS}} \sum_{i=1}^{N} \omega_i s_i - (\theta_{\text{MBS}} - \theta_{\text{SBS}}) R_{\text{SBS}} \tag{17}$$

using (15). We want to minimize the network load over the caching policies $\mu_i(t)$ under the constraints (5)–(9). The first term in (17) is independent of $\mu_i(t)$. Hence, minimizing (16) is equivalent to minimizing

$$\theta_{\text{C}} R_{\text{C}} - (\theta_{\text{MBS}} - \theta_{\text{SBS}}) R_{\text{SBS}}.$$

Thus, minimizing the network load corresponds to the optimization problem

$$\begin{aligned} &\underset{\substack{\mu_{i,j}, \nu_i \in \mathbb{R} \\ \beta_{i,j} \in \{0,1\}}}{\text{minimize}} \ \theta_{\text{C}} R_{\text{C}} - (\theta_{\text{MBS}} - \theta_{\text{SBS}}) R_{\text{SBS}}, \\ &\text{subject to (5)–(9).} \end{aligned} \tag{18}$$

We remind that optimal code parameters n_i and k_i are readily obtained from the caching policies $\mu_{i,j}$ minimizing the network load (see Section II-A). Consider briefly the case of zero cache update cost, i.e., $\theta_{\text{C}} = 0$. For $\theta_{\text{MBS}} > \theta_{\text{SBS}}$, we see that (18) represents a maximization of the SBS download rate R_{SBS} and for $\theta_{\text{MBS}} \leq \theta_{\text{SBS}}$, (18) has a trivial solution $\mu_{i,j} = 0$, i.e., caching at the SBSs is turned off ($R_{\text{SBS}} = 0$) and all data is fetched from the MBS, for which

$$W = \theta_{\text{MBS}} \sum_{i=1}^{N} \omega_i s_i$$

using (16) and (17).

Next, we reformulate the optimization problem (18) in a way that is more tractable. Using the epigraph formulation [23, Ch. 3.1.7], we introduce the auxiliary optimization variables $\xi_{b,i,j} \in \mathbb{R}$ and the constraints

$$\xi_{b,i,j} \leq 1, \tag{19}$$

$$\xi_{b,i,j} \leq b\mu_{i,j}. \tag{20}$$

Expressing the SBS download rate in (12) as

$$\tilde{R}_{\text{SBS}} = \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \xi_{b,i,j} F_{i,j}, \tag{21}$$

with the notation $\tilde{R}_{\text{SBS}}$ to emphasize that it corresponds to the download rate of the epigraph formulation, we formulate the MILP

$$\begin{aligned} &\underset{\substack{\mu_{i,j}, \nu_i, \xi_{b,i,j} \in \mathbb{R} \\ \beta_{i,j} \in \{0,1\}}}{\text{minimize}} \ \theta_{\text{C}} R_{\text{C}} - (\theta_{\text{MBS}} - \theta_{\text{SBS}}) \tilde{R}_{\text{SBS}}, \\ &\text{subject to (5)–(9), (19), (20),} \end{aligned} \tag{22}$$

which is equivalent to (18) and efficiently solved using, e.g., Gurobi [24]. The MILP (22) provides optimal FTTL caching policies for the distributed coded caching scenario. Optimal coded TTL policies are achieved by letting $\nu_i = 1$ and removing the constraint (7), while coded STTL policies are attained by removing the constraints (7)–(9). Note that the STTL optimization problem is a linear program. We observe that the optimal STTL caching policies $\mu_{i,j}$ are almost exclusively rational numbers which removes the quantization step when obtaining code parameters k_i and n_i, as explained in Section II-A.

A. Analysis as Single Cache TTL

In this subsection, we will show that the distributed coded caching problem using TTL caching policies in (18) can equivalently be analyzed as a single cache TTL problem using a particular decreasing and convex cost function. We also show how our distributed caching problem maps to the sum utility maximization (4) for the single cache case. Let the random variable Y denote the number of SBSs within range of a user, with $\Pr(Y = b) = \gamma_b$, $b = 0, 1, \ldots, B$. Changing the order of summation in (12) yields

$$R_{\text{SBS}} = \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} F_{i,j}\, \mathbb{E}[\min\{1, \mu_{i,j} Y\}]. \tag{23}$$

Regarding the expectation in (23), we will need the following lemma in subsequent theorems.

Lemma 1. *For a nonnegative random variable Y and $\mu \geq 0$,*

$$\mathbb{E}[\min\{1, \mu Y\}] = \int_0^1 1 - F_Y(z/\mu)\ \mathrm{d}z.$$

Proof: See Appendix A. ∎

The following theorem gives some important properties of the expectation in (23), as a function of the caching policy $\mu_{i,j}$.

Theorem 1. *For a nonnegative random variable Y, the expectation $\mathbb{E}[\min\{1, \mu Y\}]$ is an increasing and concave function of $\mu \geq 0$.*

Proof: See Appendix B. ∎

The result of Theorem 1 is interesting because it proves that (18) is convex. Furthermore, it shows how the cost minimization (18) with link costs $\theta_{\text{MBS}} = 1$, $\theta_{\text{SBS}} = 0$, and no cache update cost ($\theta_{\text{C}} = 0$), which corresponds to a distributed caching scenario, maps to the utility maximization (4), which assumes a single cache. The following theorem considers the important special case of SBSs distributed in an area according to a Poisson point process, in which case Y corresponds to a Poisson random variable [17, Ch. 2.3].

Theorem 2. *For $Y \sim \text{Poisson}(\lambda)$,*

$$\mathbb{E}[\min\{1, \mu Y\}] = 1 + (\lambda\mu - 1) Q(\lceil 1/\mu \rceil, \lambda) - \frac{\mathrm{e}^{-\lambda} \lambda^{\lceil 1/\mu \rceil} \mu}{\Gamma(\lceil 1/\mu \rceil)}, \tag{24}$$

where $Q(\cdot, \cdot)$ is the regularized Gamma function and $\Gamma(\cdot)$ is the Gamma function.

Proof: See Appendix C. ∎

The expression (24) is an increasing and concave function of μ, according to Theorem 1. Due to the ceiling function $\lceil 1/\mu \rceil$ in (24), we see that we should set $1/\mu \in \mathbb{N}$ in order to minimize (18) while not wasting cache capacity resources (see (5)).

V. Special Cases

The distributed coded caching problem utilizing TTL caching policies (18) has two interesting problems as special cases; static caching ($K = 0$), studied in [2], [3], [5], [6] for MDS codes, and single cache ($\gamma_1 = 1$), investigated in [16]. In this section, we show the connection between our problem and the special case of static caching, which we prove is optimal under a Poisson request process, and the special case of a single cache, which we prove has a particularly simple optimal solution.

A. Static Coded Caching

Before showing that (18) includes static caching as a special case, we have the following theorem.

Theorem 3. *For a Poisson request process, static caching minimizes* (18).

Proof: See Appendix D. ∎

Under static caching, FTTL and STTL are identical as only the updates distinguish the two caching policies. In the following, we assume that all files are of equal size, i.e., $s_i = s$, and study the nontrivial case $\theta_{\text{MBS}} > \theta_{\text{SBS}}$. For static caching ($K = 0$), (10) reduces to

$$F_{i,j} = F_{i,0} = 1 - F_{X_i}(0) = 1$$

and the objective function (18) becomes

$$\begin{aligned} -R_{\text{SBS}} &= -\sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \min\{1, b\mu_{i,0}\} F_{i,0} \\ &= -\omega s \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} p_i \min\{1, b\mu_{i,0}\}. \end{aligned} \tag{25}$$

Also, (11) is

$$A_{i,j} = A_{i,0} = \int_0^\infty 1 - F_{X_i}(t)\ \mathrm{d}t = \mathbb{E}[X_i] = \omega_i^{-1}.$$

Hence, the constraint (5) simplifies to

$$\sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \mu_{i,j} A_{i,j} = s \sum_{i=1}^{N} \mu_{i,0} \leq C. \tag{26}$$

Using (25) in (18), under constraints (6) and (26), the optimization problem (18) is precisely the static caching problem considered in [3], apart from additive and multiplicative constants. Hence, the static caching problem explored in [3] is a special case of (18).

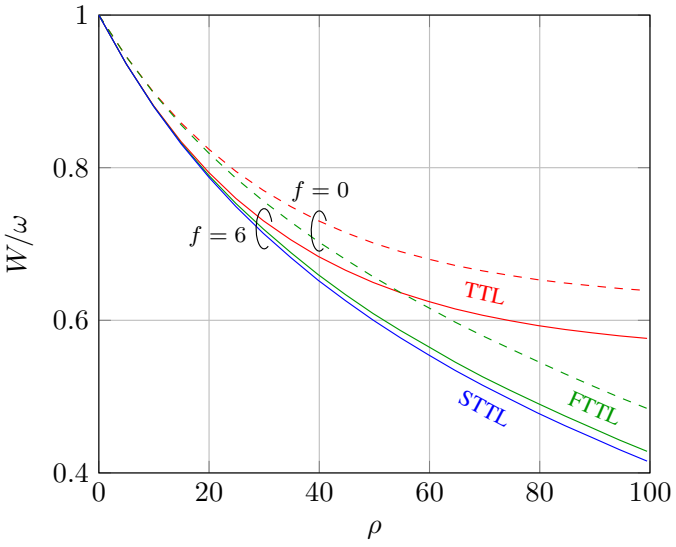

Figure 2. The fraction of data downloaded from the MBS as a function of the SBS density ρ.

B. Single Cache TTL

We proceed with the other interesting special case, i.e., the single cache problem. Letting $\gamma_1 = 1$ in (12), i.e., users access a single cache with probability 1, we see that the average rate at which data is downloaded from the SBSs is

$$R_{\text{SBS}} = \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \min\{1, \mu_{i,j}\} F_{i,j} = \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \mu_{i,j} F_{i,j}, \quad (27)$$

since $\mu_{i,j} \leq 1$ using (6). For the nontrivial case of $\theta_{\text{MBS}} > \theta_{\text{SBS}}$, and update cost $\theta_{\text{C}} = 0$, the minimization problem (18) is equivalent to a maximization problem of the objective function (27). In particular, the STTL problem has a surprisingly simple solution given by the following theorem.

Theorem 4. *For users accessing a single cache, link costs $\theta_{MBS} > \theta_{SBS}$ and $\theta_C = 0$, the STTL optimization problem, i.e., maximizing* (27) *under constraints* (5) *and* (6)*, is a fractional knapsack problem with a greedy optimal solution equivalent to FTTL and TTL.*

Proof: See Appendix E. ∎

A similar result was proved in [16] for a single file, i.e., $N = 1$.

VI. Numerical Results

In the following, we will assume that the times between requests are distributed according to the Weibull distribution, which has been shown to accurately estimate inter-request times of, e.g., educational media [18], i.e., $X_i \sim \text{Weibull}(a_i, b_i)$, where a_i, $0 < a_i \leq 1$, and b_i are the shape and scale parameters of the distribution, respectively. For simplicity, we let $a_i = a$ for all i to evaluate the network load for a class of files requested with the same Weibull shape statistics. However, we remark that our optimization framework can handle general a_i. The CDF of the Weibull distribution is

$$F_{X_i}(t) = 1 - \exp\left[-\left(\frac{t}{b_i}\right)^a\right].$$

Also,

$$\omega_i^{-1} = \mathbb{E}[X_i] = b_i \Gamma(1 + a^{-1}),$$

which implies

$$b_i = \frac{1}{\omega_i \Gamma(1 + a^{-1})}.$$

We assume the aggregate request rate per hour $\omega = 100$ and

$$p_i = \frac{1/i^\alpha}{\sum_{\ell=1}^{N} 1/\ell^\alpha},$$

which is the Zipf probability mass function with parameter $\alpha \geq 0$. We remind that p_i has the interpretation of file popularity under a Poisson request process.

The area is defined by the communication range of the MBS, which is denoted by r_{MBS} and assumed to be $r_{\text{MBS}} = 800$ meters (m), i.e., the considered area is $\pi r_{\text{MBS}}^2 \approx 2$ square kilometers. The SBSs are deployed in the area according to a Poisson point process. Let ρ be the density of SBSs per square kilometer (km^{-2}), i.e., $\rho = B/(\pi r_{\text{MBS}}^2)$. The probability that a user is within range of b SBSs is [17, Ch. 2.3]

$$\gamma_b = \mathrm{e}^{-\lambda} \frac{\lambda^b}{b!},$$

where $\lambda = \rho \pi r_{\text{SBS}}^2 = B(r_{\text{SBS}}/r_{\text{MBS}})^2$. Unless stated otherwise, we will assume the following setup for the remainder of this section. The library holds $N = 100$ files, each of normalized size $s_i = 1$. We set the Weibull shape $a = 0.6$, which describes a quite bursty request process and is within the range specified in [18]. Also, we set $\alpha = 0.7$, which has been shown to accurately capture the popularity of Youtube videos [25]. We assume that there are $B = 100$ SBSs in the area, corresponding to a density $\rho \approx 50$ km^{-2} and that users can download content from SBSs within a range of $r_{\text{SBS}} = 100$ m. Each SBS has the capacity to cache $C = 10$ files or 10% of the file library. We assume the link costs $\theta_{\text{SBS}} = 0$ and $\theta_{\text{MBS}} = 1$. Furthermore, we assume that $\theta_{\text{C}} \ll \theta_{\text{MBS}}$, which is a reasonable assumption since data can be transmitted to caches over high capacity fiber-optical or highly directional wireless backhaul links, while the MBS serves a large number of users over potentially large distances. Finally, we consider an update window length of $K/f = 1$ hour and update frequencies $f = 6$ per hour.

We obtain optimal TTL, FTTL, and STTL caching policies by solving (22) and plot the network load (16) normalized by the aggregate request rate ω. For $\theta_{\text{C}} = 0$, the network load is interpreted as the fraction of data downloaded from the MBS. Fig. 2 shows this fraction as a function of the SBS density ρ for no cache updates, i.e., $f = 0$ implying $\mu_{i,j} = \mu_{i,0}$, and cache update frequency per hour $f = 6$. The network load using FTTL and STTL overlap for $f = 0$, which is expected since only the cache updates distinguish the two policies. We also see that there is a reduction in network load when choosing the FTTL or STTL caching policies over the TTL policy and that

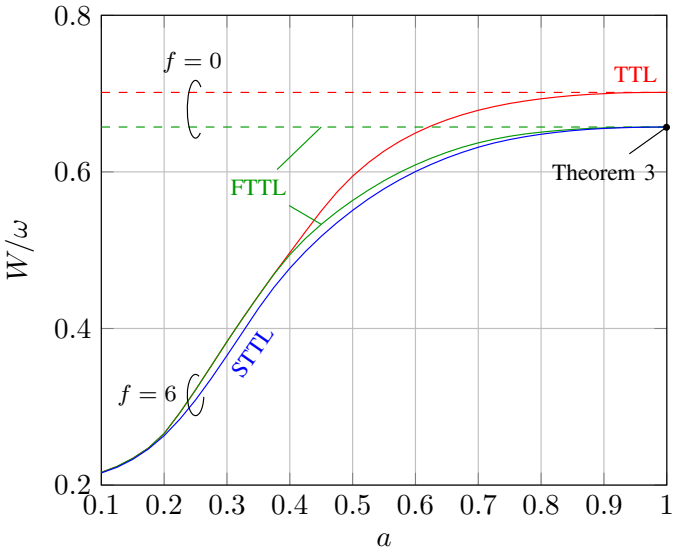

Figure 3. The fraction of data downloaded from the MBS as a function of the Weibull shape a for an update frequency $f = 6$.

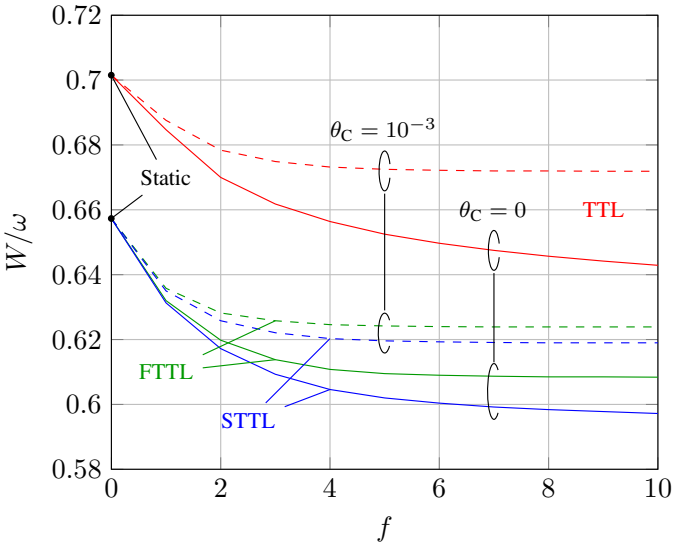

Figure 4. Normalized network load as a function of the cache update frequency f.

the network load decreases with increasing SBS density. The reason for the performance loss when using the TTL caching policy is that users within range of $b > 1$ SBSs will download superfluous data, which correspond to a wasteful use of cache memory resources. The gain for the static caching scenario ($f = 0$) was observed already in [3]. Finally, we observe that, for $f = 6$, there is only a small reduction in network load for STTL as compared to FTTL, but the load reduction is increasing for increasing ρ.

Fig. 3 shows the fraction of data downloaded from the MBS versus the Weibull shape a, for update frequency per hour $f = 6$, and no cache update cost ($\theta_C = 0$). We also include curves for static caching ($f = 0$), which do not depend on a as is shown in (25) and (26), for comparison. For bursty request arrivals, i.e., small values of a, we see that the use of TTL caching policies reduces the fraction of data downloaded from the MBS significantly with respect to static caching. Furthermore, we observe that, for very small values of a, all TTL policies have similar performance. This is because, with high probability, the times between requests are less than the period T, i.e., $F_{i,0} \approx 1$ for all i, and TTL is an optimal caching policy. For $a = 1$, corresponding to a Poisson request process, FTTL and STTL yield the same network load as proved in Theorem 3, which is, however, lower than the network load using TTL. A similar effect was shown in [16] for the single cache case.

Fig. 4 shows the normalized minimum network load as a function of the update frequency f for the case of no cache update cost ($\theta_C = 0$) and $\theta_C = 10^{-3}$. For both cases, updating content on the caches is seen to be beneficial for all TTL policies. For example, using STTL and assuming $\theta_C = 0$, the reduction is roughly 10% as compared to static caching. We observe that the decrease in network load for an increase in update frequency saturates for moderately large f. Hence, cache updates need not be very frequent to reap the benefits of the TTL, FTTL, and STTL caching policies. The sufficient update frequency of course depends on several parameter values, in particular, the Weibull shape a is a key parameter when deciding update frequencies.

Finally, in Fig. 5, the normalized minimum network load is plotted versus the cache update cost θ_C for an update frequency per hour $f = 6$. The load for static caching ($f = 0$) is also shown in the figure. As previously described, the network load when using FTTL and STTL is the same for static caching. It is interesting to note that all TTL policies revert to static caching for sufficiently large values of θ_C. Also included in Fig. 5 is a simulation of the optimal (under synchronous cache updates) STTL and TTL caching policies for asynchronous cache updates, i.e., only the SBSs within range of a user placing a request update cached content. The considered caching policies do not exhibit a better performance under asynchronous updates, which is to be expected for two reasons. Firstly, since the file request process is homogenous over the considered area, the spatial average cached content is important and the same average cached content can be achieved by both synchronous and asynchronous cache updates. Secondly, the request rate within the communication range of an SBS is smaller than the request rate in the entire area, implying less content to be cached over time using asynchronous updates, i.e., the caches are underutilized.

VII. Conclusion

We optimized time-to-live (TTL) caching policies with periodic eviction of coded content to minimize the overall network load for a scenario where content is encoded using a maximum distance separable (MDS) code and cached in a distributed fashion across small base stations. The proposed optimization problem is efficiently solved numerically. Interestingly, we show that the problem can equivalently be analyzed as a single cache optimization problem under a specific decreasing and convex cost function. For small base stations (SBSs) deployed according to a Poisson point process, we provide the cost function explicitly. The analyzed scenario encompasses static caching and single caching as important special cases. We

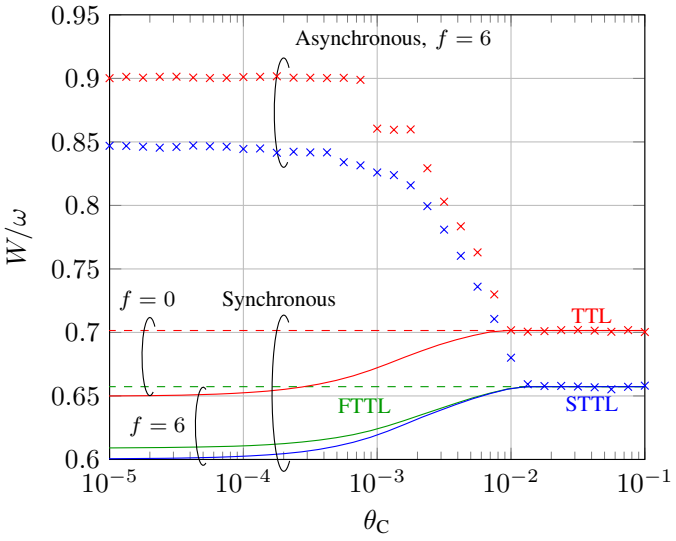

Figure 5. Normalized network load versus the update cost θ_{C} when using the various caching policies under synchronous and asynchronous cache updates.

proved that, interestingly, static caching is optimal under a Poisson request process. We also proved that the single cache problem has a simple greedy solution. We showed that TTL caching policies can offer substantial reductions in network load compared with static caching under a request renewal process, in particular when the request process is bursty. Conversely, for sufficiently large cache update cost, dynamic caching of MDS coded content is futile, i.e., static caching is optimal. Finally, although we consider a wireless network scenario, the results are general in the sense that they can be applied to any distributed caching scenario.

The updating of coded content cached at SBSs can be seen as content repair [4]. Therefore, an interesting question is whether repair-efficient coding schemes, e.g., regenerating codes [26], can yield a lower network load.

Appendix A
Proof of Lemma 1

We represent 1 as a random variable with degenerate distribution $\delta(z-1)$, where $\delta(\cdot)$ is the Dirac delta function, and let $Z = \min\{1, \mu Y\}$, for which the CDF of Z is

$$\begin{aligned} F_Z(z) &\triangleq \Pr(Z \le z) \\ &= \Pr(\min\{1, \mu Y\} \le z) \\ &= 1 - \Pr(1 > z, \mu Y > z) \\ &= 1 - H(1-z)(1 - F_Y(z/\mu)), \end{aligned}$$

where $H(\cdot)$ is the heavyside function. The expected value of Z is

$$\begin{aligned} \mathbb{E}[Z] &= \int_0^\infty 1 - F_Z(z)\ \mathrm{d}z \\ &= \int_0^\infty H(1-z)(1 - F_Y(z/\mu))\ \mathrm{d}z \\ &= \int_0^1 1 - F_Y(z/\mu)\ \mathrm{d}z. \end{aligned}$$

Appendix B
Proof of Theorem 1

Since $F_Y(y)$ is an increasing function of y, $1 - F_Y(z/\mu)$ is an increasing function of μ, and

$$\int_0^1 1 - F_Y(z/\mu)\ \mathrm{d}z$$

is an increasing function of μ. Using Lemma 1, the expectation $\mathbb{E}[\min\{1, \mu Y\}]$ is an increasing function of μ.

For $\mu_1 \ge 0$, $\mu_2 \ge 0$, $Y \ge 0$, and $0 \le \alpha \le 1$, the following inequalities hold,

$$\begin{aligned} (1-\alpha)\mu_1 Y &\ge (1-\alpha)\min\{1, \mu_1 Y\}, \\ \alpha\mu_2 Y &\ge \alpha \min\{1, \mu_2 Y\}. \end{aligned}$$

Hence,

$$((1-\alpha)\mu_1 + \alpha\mu_2)Y \ge (1-\alpha)\min\{1, \mu_1 Y\} + \alpha\min\{1, \mu_2 Y\}. \tag{28}$$

Similarly, using

$$\begin{aligned} (1-\alpha) &\ge (1-\alpha)\min\{1, \mu_1 Y\}, \\ \alpha &\ge \alpha \min\{1, \mu_2 Y\}, \end{aligned}$$

we have that

$$1 = 1 - \alpha + \alpha \ge (1-\alpha)\min\{1, \mu_1 Y\} + \alpha\min\{1, \mu_2 Y\}. \tag{29}$$

Using (28) and (29), we get

$$\begin{aligned} &\min\{1, ((1-\alpha)\mu_1 + \alpha\mu_2)Y\} \\ &\qquad \ge (1-\alpha)\min\{1, \mu_1 Y\} + \alpha\min\{1, \mu_2 Y\}. \end{aligned}$$

Taking the expectation of both sides yields

$$\begin{aligned} &\mathbb{E}[\min\{1, ((1-\alpha)\mu_1 + \alpha\mu_2)Y\}] \\ &\qquad \ge \mathbb{E}[(1-\alpha)\min\{1, \mu_1 Y\} + \alpha\min\{1, \mu_2 Y\}] \\ &\qquad = (1-\alpha)\,\mathbb{E}[\min\{1, \mu_1 Y\}] + \alpha\,\mathbb{E}[\min\{1, \mu_2 Y\}], \end{aligned}$$

which concludes the proof.

Appendix C
Proof of Theorem 2

For a Poisson random variable Y with rate λ,

$$F_Y(y) = \sum_{i=0}^{\lfloor y \rfloor} \frac{\lambda^i}{i!}\mathrm{e}^{-\lambda}. \tag{30}$$

For a positive integer x, let $\Gamma(x)$ and $Q(x, \lambda)$ denote the Gamma function and the regularized Gamma function, i.e., $\Gamma(x) = (x-1)!$ and

$$Q(x+1, \lambda) = \int_\lambda^\infty \frac{t^x \mathrm{e}^{-t}}{\Gamma(x+1)}\ \mathrm{d}t \overset{(a)}{=} \frac{\lambda^x \mathrm{e}^{-\lambda}}{\Gamma(x+1)} + Q(x, \lambda), \tag{31}$$

respectively, where (a) is obtained after integration by parts. Unfolding the recursion in (31), using $Q(1, \lambda) = \mathrm{e}^{-\lambda}$ yields

$$Q(x+1, \lambda) = \sum_{i=0}^{x} \frac{\lambda^i}{\Gamma(i+1)}\mathrm{e}^{-\lambda} = \sum_{i=0}^{x} \frac{\lambda^i}{i!}\mathrm{e}^{-\lambda} \tag{32}$$

and

$$Q(\lfloor y \rfloor + 1, \lambda) = F_Y(y), \tag{33}$$

using (30).

Using (33),

$$\int_0^1 F_Y(y/\mu)\ \mathrm{d}y = \int_0^1 Q(\lfloor y/\mu \rfloor + 1, \lambda)\ \mathrm{d}y = \mu \sum_{i=1}^{\lfloor 1/\mu \rfloor} Q(i,\lambda) + (1 - \lfloor 1/\mu \rfloor \mu) Q(\lfloor 1/\mu \rfloor + 1, \lambda), \tag{34}$$

where the integral is a summation due to the floor function in the argument of $Q(\cdot,\cdot)$. Applying the recursion (31) repeatedly yields

$$\begin{aligned}\sum_{i=1}^{\lfloor 1/\mu \rfloor} Q(i,\lambda) &= \lfloor 1/\mu \rfloor Q(\lfloor 1/\mu \rfloor + 1, \lambda) - \sum_{i=1}^{\lfloor 1/\mu \rfloor} i \frac{\lambda^i \mathrm{e}^{-\lambda}}{\Gamma(i+1)} \\ &= \lfloor 1/\mu \rfloor Q(\lfloor 1/\mu \rfloor + 1, \lambda) - \lambda \sum_{j=0}^{\lfloor 1/\mu \rfloor - 1} \frac{\lambda^j \mathrm{e}^{-\lambda}}{\Gamma(j+1)} \\ &\overset{(b)}{=} \lfloor 1/\mu \rfloor Q(\lfloor 1/\mu \rfloor + 1, \lambda) - \lambda Q(\lfloor 1/\mu \rfloor, \lambda),\end{aligned}$$

where we have used (32) in (b). Inserting this expression in (34), one obtains

$$\begin{aligned}\int_0^1 F_Y(y/\mu)\ \mathrm{d}y &= Q(\lfloor 1/\mu \rfloor + 1, \lambda) - \mu\lambda Q(\lfloor 1/\mu \rfloor, \lambda) \\ &\overset{(c)}{=} (1 - \lambda\mu) Q(\lceil 1/\mu \rceil, \lambda) + \frac{\lambda^{\lceil 1/\mu \rceil} \mu \mathrm{e}^{-\lambda}}{\Gamma(\lceil 1/\mu \rceil)},\end{aligned} \tag{35}$$

where we have used (31) and

$$\lceil 1/\mu \rceil - \lfloor 1/\mu \rfloor = \begin{cases} 0, & \text{if } 1/\mu \in \mathbb{Z} \\ 1, & \text{if } 1/\mu \notin \mathbb{Z} \end{cases}$$

in (c). Combining (35) with the result of Lemma 1 yields the desired result.

Appendix D
Proof of Theorem 3

For proving that static caching minimizes (22) (and hence (18)) it is sufficient to show that it maximizes $\tilde{R}_{\text{SBS}}$ (see (21)), as for static caching $R_C = 0$. Maximizing (21) under constraints (5), (6), (19), and (20), is equivalent to

$$\underset{\mu_{i,j}, \xi_{b,i,j}, C_i \in \mathbb{R}}{\text{maximize}} \sum_{b=0}^{B} \gamma_b \sum_{i=1}^{N} \omega_i s_i \sum_{j=0}^{K} \xi_{b,i,j} F_{i,j}, \tag{36}$$

$$\text{subject to } s_i \sum_{j=0}^{K} \mu_{i,j} F_{i,j} \le C_i, \tag{37}$$

$$\sum_{i=1}^{N} C_i = C, \tag{38}$$

$$\mu_{i,j} - \mu_{i,j-1} \le 0,\ \mu_{i,-1} = 1, \tag{39}$$

$$-\mu_{i,j} \le 0, \tag{40}$$

$$\xi_{b,i,j} \le 1, \tag{41}$$

$$\xi_{b,i,j} \le b\mu_{i,j}, \tag{42}$$

where C_i may be regarded as the size of the cache partition reserved for file i, and we used the fact that for exponentially distributed inter-request times $F_{i,j}/A_{i,j} = \omega_i$ (from (10) and (11)), i.e., the hazard function is constant for a Poisson request process. For fixed C_i's, the maximization problem is separable in i. Thus, we can consider the following optimization problem

$$\underset{\mu_{i,j}, \xi_{b,i,j} \in \mathbb{R}}{\text{maximize}} \sum_{b=0}^{B} \gamma_b \sum_{j=0}^{K} \xi_{b,i,j} F_j, \tag{43}$$

$$\text{subject to } s_i \sum_{j=0}^{K} \mu_{i,j} F_{i,j} \le C_i, \tag{44}$$

$$\mu_{i,j} - \mu_{i,j-1} \le 0,\ \mu_{i,-1} = 1, \tag{45}$$

$$-\mu_{i,j} \le 0, \tag{46}$$

$$\xi_{b,i,j} \le 1, \tag{47}$$

$$\xi_{b,i,j} \le b\mu_{i,j}, \tag{48}$$

for each file $i = 1, \ldots, N$ separately. We can now prove the following lemma.

Lemma 2. *Static caching is an optimal solution to* (43)–(48).

Proof: For ease of exposition, we drop the subindex i in the proof. Introducing the dual variables λ, ϕ_j, ψ_j, $\delta_{b,j}$, and $\epsilon_{b,j}$, the Karush-Kuhn-Tucker (KKT) conditions of (43)–(48) are

$$-\gamma_b F_j + \delta_{b,j} + \epsilon_{b,j} = 0, \tag{49}$$

$$\lambda s F_j + \phi_j - \phi_{j+1} - \psi_j - \sum_{b=0}^{B} \epsilon_{b,j} b = 0,\ \phi_{K+1} = 0, \tag{50}$$

$$\lambda \left(-C + s \sum_{j=0}^{K} \mu_j F_j \right) = 0, \tag{51}$$

$$\phi_j(\mu_j - \mu_{j-1}) = 0,\ \mu_{-1} = 1, \tag{52}$$

$$\psi_j(-\mu_j) = 0, \tag{53}$$

$$\delta_{b,j}(\xi_{b,j} - 1) = 0, \tag{54}$$

$$\epsilon_{b,j}(\xi_{b,j} - b\mu_j) = 0, \tag{55}$$

$$\lambda \ge 0, \phi_j \ge 0, \psi_j \ge 0, \tag{56}$$

$$\delta_{b,j} \ge 0, \epsilon_{b,j} \ge 0, \tag{57}$$

and (44)–(48). Let

$$\mu_j = C/s,$$
$$\xi_{b,j} = \min\{1, bC/s\},$$

which corresponds to static caching utilizing completely the given cache partition, and largest possible values of the variables $\xi_{b,j}$. Furthermore, let

$$\phi_j = 0, \tag{58}$$

$$\psi_j = 0, \tag{59}$$

$$\delta_{b,j} = \begin{cases} 0, & \text{if } b \le s/C \\ \gamma_b F_j, & \text{if } b > s/C \end{cases}, \tag{60}$$

$$\epsilon_{b,j} = \begin{cases} \gamma_b F_j, & \text{if } b \le s/C \\ 0, & \text{if } b > s/C \end{cases}, \tag{61}$$

$$\lambda = \frac{1}{s} \sum_{j=0}^{K} \sum_{b=0}^{B} \epsilon_{b,j} b \overset{(a)}{=} \frac{1}{s} \sum_{b=0}^{\lfloor s/C \rfloor} \gamma_b b, \tag{62}$$

where we have used (61) and (14) in (a). It is readily verified that the choice of optimization and dual variables satisfy the KKT conditions and are hence optimal since the problem is convex [23, Ch. 5.5.3]. Therefore, static caching maximizes (43). ∎

It remains to optimize over the C_i, but since, by Lemma 2, static caching is optimal for any assignment of C_i's, it is optimal for (36)–(42).

Appendix E
Proof of Theorem 4

Letting $\gamma_1 = 1$, $\theta_{\text{C}} = 0$, and $\theta_{\text{MBS}} > \theta_{\text{SBS}}$, the STTL problem, i.e., maximizing (27) under constraints (5) and (6), is equivalent to

$$\underset{\mu_{i,j}\in\mathbb{R}}{\text{maximize}} \ \sum_{i=1}^{N}\omega_i s_i \sum_{j=0}^{K}\mu_{i,j}F_{i,j}, \tag{63}$$

$$\text{subject to} \ \sum_{i=1}^{N}\omega_i s_i \sum_{j=0}^{K}\mu_{i,j}A_{i,j} \leq C, \tag{64}$$

$$1 \geq \mu_{i,0} \geq \mu_{i,1} \geq \cdots \geq \mu_{i,K} \geq 0. \tag{65}$$

Relaxing the constraint (65), replacing it with $0 \leq \mu_{i,j} \leq 1$, and letting $x_{i,j} = \omega_i s_i \mu_{i,j} A_{i,j}$, we obtain

$$\underset{x_{i,j}\in\mathbb{R}}{\text{maximize}} \ \sum_{i=1}^{N}\sum_{j=0}^{K} x_{i,j}\frac{F_{i,j}}{A_{i,j}},$$

$$\text{subject to} \ \sum_{i=1}^{N}\sum_{j=0}^{K} x_{i,j} \leq C,$$

$$0 \leq x_{i,j} \leq \omega_i s_i A_{i,j},$$

which is recognized as the fractional knapsack problem [27]. The optimal solution to this problem is obtained by setting $x_{i,j} = \omega_i s_i A_{i,j}$, i.e., $\mu_{i,j} = 1$, greedily with respect to the fractions $F_{i,j}/A_{i,j}$ [27]. We observe that, since $F_{i,j}/A_{i,j}$ is decreasing in j as explained in Sec. III, the constraint (65) is met and we have a valid STTL caching policy. Apart from the possibility that one $\mu_{i,j} < 1$ depending on the value of C, the constraints (7)–(9) are also satisfied by this policy and, hence, the optimal STTL caching policy is equivalent to the optimal FTTL and TTL caching policies.

References

[1] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[2] 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.

[3] V. Bioglio, F. Gabry, and I. Land, "Optimizing MDS codes for caching at the edge," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, 2015.

[4] J. Pedersen, A. Graell i Amat, I. Andriyanova, and F. Brännström, "Distributed storage in mobile wireless networks with device-to-device communication," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Nov. 2016.

[5] ——, "Optimizing MDS coded caching in wireless networks with device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 286–295, Jan. 2019.

[6] 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.

[7] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[8] M. Ji, G. Caire, and A. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[9] Y. Wei and S. Ulukus, "Novel decentralized coded caching through coded prefetching," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Kaohsiung, Taiwan, 2017.

[10] H. Reisizadeh, M. A. Maddah–Ali, and S. Mohajer, "Erasure coding for decentralized coded caching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, 2018, pp. 1715–1719.

[11] E. Gelenbe, "A unified approach to the evaluation of a class of replacement algorithms," *IEEE Trans. Comput.*, vol. C-22, pp. 611–618, Jun. 1973.

[12] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.

[13] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. 24th Int. Teletraffic Congr.*, Kraków, Poland, 2012.

[14] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in LRU caching?" *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 59–67, Jul. 2013.

[15] M. Dehghan, L. Massoulié, D. Towsley, D. S. Menasché, and Y. C. Tay, "A utility optimization approach to network cache design," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1013–1027, Jun. 2019.

[16] J. Goseling and O. Simeone, "Soft-TTL: Time-varying fractional caching," *IEEE Netw. Lett.*, vol. 1, no. 1, pp. 18–21, Mar. 2019.

[17] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*. Wiley, 2013.

[18] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto, "Analyzing client interactivity in streaming media," in *Proc. 13th Int. Conf. World Wide Web*, New York, NY, 2004, pp. 534–543.

[19] L. Chen, L. Song, J. Chakareski, and J. Xu, "Collaborative content placement among wireless edge caching stations with time-to-live cache," *IEEE Trans. Multimedia*, vol. 22, no. 2, pp. 432–444, Feb. 2019.

[20] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Vancouver, BC, 2002, pp. 271–280.

[21] E. Recayte, F. Lázaro, and G. Liva, "Caching at the edge with LT codes," in *Proc. 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Hong Kong, China, 2018.

[22] G. Neglia, D. Carra, and P. Michiardi, "Cache policies for linear utility maximization," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 302–313, Feb. 2018.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[24] Gurobi Optimization, LLC. Gurobi optimizer reference manual. [Online]. Available: https://www.gurobi.com

[25] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Proc. 16th Int. Workshop Quality Service*, Enschede, The Netherlands, 2008.

[26] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[27] G. B. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. 266–288, Apr. 1957.