A TTI-based Approach for Content Placement in Edge Networks
Nitish K. Panigrahy*, Jian Li*, Member, IEEE, Faheem Zafari, Student Member, IEEE, Don Towsley, Life Fellow, IEEE, and Paul Yu, Member, IEEE

Abstract—Edge networks are promising to provide better services to users by provisioning computing and storage resources at the edge of networks. However, due to the uncertainty and diversity of user interests, content popularity, distributed network structure, cache sizes, it is challenging to decide where to place the content, and how long it should be cached. In this paper, we study the utility optimization of content placement at edge networks through timer-based (TTI) policies. We propose provably optimal distributed algorithms that operate at each network cache to maximize the overall network utility. Our TTI-based optimization model provides theoretical answers to how long each content must be cached, and where it should be placed in the edge network. Extensive evaluations show that our algorithm significantly outperforms path replication with conventional caching algorithms over some network topologies.

Index Terms—TTI Cache; Utility Maximization; Distributed Algorithms; Cost Minimization; Edge Network

I. INTRODUCTION

Content distribution has become a dominant application in today’s Internet. Much of these contents are delivered by Content Distribution Networks (CDNs), which are provided by Akamai, Amazon, etc. [1]. There usually exists a stringent requirement on the latency between the service provider and end users for these applications. CDNs use a large network of caches to deliver content from a location close to the end users. This aligns with the trend of edge networks, where computing and storage resources are provisioned at the edge of networks. If a user’s request is served by a nearby edge cache, the user experiences a faster response time than if it was served by the backend server. It also reduces bandwidth requirements at the central content repository.

With the aggressive increase in Internet traffic over past years [2], CDNs need to host content from thousands of regions belonging to web sites of thousands of content providers. Furthermore, each content provider may host a large variety of content, including videos, music, and images. Such an increasing diversity in content services requires CDNs to provide different quality of service to varying content classes and applications with different access characteristics and performance requirements. Significant economic benefits and important technical gains have been observed with the deployment of service differentiation [3]. While a rich literature has studied the design of fair and efficient caching algorithms for content distribution, little work has paid attention to the provision of multi-level services in edge networks.

Managing edge networks requires policies to route end-user requests to the local distributed caches, as well as caching algorithms to ensure availability of requested content at the cache. In general, there are two classes of policies for studying the performance of caching algorithms: timer-based, i.e., Time-To-Live (TTI) [4], [6] and non-timer-based caching algorithms, e.g., Least-Recently-Used (LRU) [7], Least-Frequently-Used (LFU) [7], First In First Out (FIFO), and RANDOM [7]. Since the cache size is usually much smaller than the total amount of content, some contents need to be evicted if the requested content is not in the cache. Exact analysis of LRU, LFU, FIFO and RANDOM has proven to be difficult, even under the simple Independence Reference Model (IRM) [7], where requests are independent of each other. The strongly coupled nature of these eviction algorithms makes implementation of differential services challenging. In contrast, a TTI cache associates each content with a timer upon request and the content is evicted from the cache on timer expiry, independent of other contents. Analysis of these policies is simple since the eviction of contents are decoupled from each other.

Most studies have focused on the analysis of a single edge cache. When an edge network is considered, independence across different caches is usually assumed [9]. Again, it is hard to analyze most conventional caching algorithms, such as LRU, FIFO and RANDOM, but some accurate results for TTI caches are available [6], [9]. However, it has been observed [10] that performance gains can be obtained if decision-making is coupled at different caches.

In this paper, we consider a TTI-based edge network, where a set of network caches host a library of unique contents, and serve a set of users. Figure [1] illustrates an example of such a network, which is consistent with the YouTube video delivery system [11]. Each user can generate a request for a content, which is forwarded along a fixed path from the edge cache towards the server. Forwarding stops upon

*Authors with equal contribution.

This research was sponsored by the U.S. ARL and the U.K. MoD under Agreement Number W911NF-16-3-0001 and by the NSF under Grant CNS-1617437. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

N. Panigrahy and D. Towsley are with the University of Massachusetts, Amherst, MA, 01003, USA (e-mail: {nitish, towsley}@cs.umass.edu).
J. Li is with Binghamton University, the State University of New York, Binghamton, NY, 13902, USA (e-mail: lji@binghamton.edu).
F. Zafari is with Imperial College London, London, SW7 2BT, UK, (e-mail: faheem16@imperial.ac.uk).
P. Yu is with U.S. Army Research Laboratory, Adelphi, MD 20783, USA (e-mail: paul.yu.civ@mail.mil).
a cache hit, i.e., the requested content is found in a cache on the path. When such a cache hit occurs, the content is sent over the reverse path to the edge cache initializing the request. This raises the questions: where to cache the requested content on the reverse path and what is the value of its timer? Answering these questions can provide new insights in edge network design. However, it may also increase the complexity and hardness of the analysis.

Our goal is to provide thorough and rigorous answers to these questions. To that end, we consider moving the content one cache up (towards the user) if there is a cache hit on it and pushing the content one cache down (away from the user) once its timer expires in the cache hierarchy, since the recently evicted content may still be in demand. This policy is known as “Move Copy Down with Push” (MCDP) policy.

We first formulate a utility-driven caching framework for linear edge networks, where each content is associated with a utility and content is managed with a timer whose duration is set to maximize the aggregate utility for all contents over the edge network. Building on MCDP policy, we formulate the optimal TTL policy as a non-convex optimization problem in Section III. One contribution of this paper is to show that this non-convex problem can be transformed into a convex one by change of variables. We further develop distributed algorithms for content management over linear edge networks, and show that this algorithm converges to the optimal solution.

Informed by our results for linear edge networks, we consider a general cache network where each edge cache serves distinct contents, i.e., there are no common contents among edge caches, in Section IV-B. We show that this edge network can be treated as a union of different linear edge networks between each edge cache and the central server.

We further consider a more general case where common content is requested among edge caches in Section IV-C. This introduces non-convex constraints, resulting in a non-convex utility maximization problem. We show that although the original problem is non-convex, the duality gap is zero. Based on this, we design a distributed iterative primal-dual algorithm for content placement in edge network. We show through numerical evaluations that our algorithm outperforms path replication with traditional caching algorithms over a broad array of network topologies.

We provide some discussions on extension of our utility maximization framework under MCDP to a general graph based cache networks in Section V. We show how our framework can be directly mapped to content distributions in CDNs, ICNs/CCNs etc. Numerical results are given on how to optimize the performance. Also, since we consider an edge network, content placement induces costs, such as search cost for finding the requested content on the path, fetch cost to serve the content to the user that requested it, and move cost upon cache hit or miss due to caching policy. We fully characterize these costs and formulate a cost minimization problem in Section VI. We discuss related works in Section VII and conclude the paper in Section VIII.

II. PRELIMINARIES

We represent the edge cache network shown in Figure 1 by a graph \( G = (V, E) \). We use \( D = \{d_1, \ldots, d_n\} \) with \( |D| = n \) to denote the set of contents. Each network cache \( v \in V \) can store up to \( B_v \) contents to serve requests from users. We assume that each user will first send a request for the content to its local network cache, which may then route the request to other caches for retrieving the content. Without loss of generality, we assume that there is a fixed and unique path from the local cache towards a terminal cache that is connected to a server that always contains the content.

To be more specific, a request \((v, i, p)\) is determined by the local cache, \( v \), that firstly received the user request, the requested content, \( i \), and the path, \( p \), over which the request is routed. We denote a path \( p \) of length \(|p| = L\) as a sequence \( \{v_{1p}, v_{2p}, \ldots, v_{Lp}\} \) of nodes \( v_{ip} \in V \) such that \( (v_{ip}, v_{(i+1)p}) \in E \) for \( l \in \{1, \ldots, L\} \), where \( v_{Lp} = v \). We assume that path \( p \) is loop-free and terminal cache \( v_{1p} \) is the only cache on path \( p \) that accesses the server for content \( i \).

We assume that the request processes for distinct contents are described by independent Poisson processes with arrival rate \( \lambda_i \), for content \( i \in D \). Denote \( \Lambda = \sum_{i=1}^{n} \lambda_i \). Then the popularity (request probability) of content \( i \) satisfies \( 13 \):

\[
\rho_i = \frac{\lambda_i}{\Lambda}, \quad i = 1, \ldots, n. \tag{1}
\]

We consider TTL caches in this paper. Each content \( i \) is associated with a timer \( T_{ij} \) at cache \( j \). Suppose content \( i \) is requested and routed along path \( p \). There are two cases: (i) content \( i \) is not in any cache along path \( p \), in which case content \( i \) is fetched from the server and inserted into the terminal cache (denoted by cache 1). Its timer is set to \( T_{ij} \); (ii) if content \( i \) is in cache \( l \) along path \( p \), content \( i \) is moved to cache \( l + 1 \) preceding cache \( l \) in which \( i \) is found, and the timer at cache \( l + 1 \) is set to \( T_{ij(i+1)} \). Content \( i \) is pushed one cache back to cache \( l - 1 \) and the timer is set to \( T_{ij(i-1)} \) once the timer expires. We call it Move Copy Down with Push (MCDP) \( 14 \). Denote the hit probability of content \( i \) as \( h_i \), then the corresponding hit rate is \( \lambda_i h_i \).

\(^1\)Since we consider path \( p \), for simplicity, we move the dependency on \( p \) and \( v \), denote it as nodes 1, \( \cdots \), \( L \) directly.
Denote by \( \mathcal{P} \) the set of all requests, and \( \mathcal{P}_i \) the set of requests for content \( i \). Suppose a network cache \( v \) serves two requests \((v_1, i_1, p_1)\) and \((v_2, i_2, p_2)\), then there are two cases: (i) non-common requested content, i.e., \( i_1 \neq i_2 \); and (ii) common requested content, i.e., \( i_1 = i_2 \). In the following, we will focus on how to design optimal TTL policies for content placement in an edge cache network under these two cases.

While classical cache eviction policies such as LRU provide good performance and are easy to implement, Garetto et al. [15] showed that K-LRU\(^2\) can provide significant improvements over LRU even for very small K. Furthermore, Ramadan et al. [11] proposed K-LRU with big cache abstraction (K-LRU(B)) to effectively utilize resources in a hierarchical network of cache servers. Thus, in the rest of the paper, we compare the performance of MCDP to K-LRU and K-LRU(B).

III. LINEAR EDGE NETWORK

We begin with a linear edge network, i.e., there is a single path between the user and the server, composed of \( |p| = L \) caches labeled \( 1, \ldots, L \). A content enters the edge network via cache 1, and is promoted to a higher index cache whenever a cache hit occurs. In the following, we consider the MCDP replication strategy when each cache operates with a TTL policy. We also consider another replication strategy “Move Copy Down” (MCD) and relegate our discussion on it to Appendix B and Appendix C.

A. Stationary Behavior

Requests for content \( i \) arrive according to a Poisson process with rate \( \lambda_i \). Under TTL, content \( i \) spends a deterministic time in a cache if it is not requested, independent of all other contents. We denote the timer as \( T_{il} \) for content \( i \) in cache \( l \) on the path \( p \), where \( l \in \{1, \ldots, |p|\} \).

Denote by \( \mathcal{h}_l^{k} \) the \( k \)-th time that content \( i \) is either requested or the timer expires. For simplicity, we assume that content is in cache 0 (i.e., server) when it is not in the cache network. We then define a discrete time Markov chain (DTMC) \( \{X_k^l\}_{k \geq 0} \) with \( |p| + 1 \) states, where \( X_k^l \) is the index of the cache that content \( i \) is in at time \( t_k^l \). The event that the time between two requests for content \( i \) exceeds \( T_{il} \) occurs with probability \( e^{-\lambda_i T_{il}} \); consequently we obtain the transition probability matrix of \( \{X_k^l\}_{k \geq 0} \) and compute the stationary distribution. The timer-average probability that content \( i \) is in cache \( l \in \{1, \ldots, |p|\} \)

\[
\begin{align*}
\mathcal{h}_{il} &= \frac{e^{\lambda_i T_{il}} - 1}{1 + \sum_{j=1}^{\lceil |p| \rceil} (e^{\lambda_i T_{il}} - 1) \cdots (e^{\lambda_j T_{il}} - 1)}, \\
\mathcal{h}_{il} &= \frac{e^{\lambda_i T_{il} - 1}}{1 + \sum_{j=1}^{\lceil |p| \rceil} (e^{\lambda_i T_{il} - 1}) \cdots (e^{\lambda_j T_{il} - 1})}, \quad l = 2, \ldots, |p|,
\end{align*}
\]

where \( \mathcal{h}_{il} \) is also the hit probability for content \( i \) at cache \( l \).

Remark 1. The stationary analysis of MCDP is similar to a different caching policy LRU(m) considered in [16]. We relegate its explicit expression to Appendix A and also refer interested readers to [16] for more detail.

B. From Timer to Hit Probability

We consider a TTL cache network where requests for different contents are independent of each other and each content \( i \) is associated with a timer \( T_{il} \) at each cache \( l \in \{1, \ldots, |p|\} \) on the path. Denote \( T_i = (T_{i1}, \ldots, T_{ip}) \) and \( T = (T_1, \ldots, T_n) \). From (2), the overall utility on path \( p \) is given as

\[
\sum_{i \in \mathcal{D}} \sum_{l=1}^{|p|} \psi_i |p| - l U_i (\lambda_i, h_{il}(T)) ,
\]

where the utility function \( U_i : [0, \infty) \rightarrow \mathbb{R} \) is assumed to be increasing, continuously differentiable, and strictly concave function of content hit rate, and \( 0 < \psi_i \leq 1 \) is a discount factor capturing the utility degradation along the request’s routing direction. Since each cache is finite in size, we have the capacity constraint

\[
\sum_{i \in \mathcal{D}} h_{il}(T) \leq B_l, \quad l \in \{1, \ldots, |p|\} ,
\]

Therefore, the optimal TTL policy for content placement on path \( p \) is the solution of the following optimization problem

\[
\begin{align*}
\max_T & \sum_{i \in \mathcal{D}} \sum_{l=1}^{|p|} \psi_i |p| - l U_i (\lambda_i, h_{il}(T)) \\
\text{s.t.} & \sum_{i \in \mathcal{D}} h_{il}(T) \leq B_l, \quad l \in \{1, \ldots, |p|\} , \quad T_{il} \geq 0, \quad \forall i \in \mathcal{D}, \quad l = 1, \ldots, |p| ,
\end{align*}
\]

where \( h_{il}(T) \) is given in (3). However, (5) is a non-convex optimization with a non-linear constraint. Our objective is to characterize the optimal timers for different contents on path \( p \). To that end, it is helpful to express (5) in terms of hit probabilities. In the following, we discuss how to change the variables from timer to hit probability.

Since \( 0 \leq T_{il} \leq \infty \), it is easy to check that \( 0 \leq h_{il} \leq 1 \) for \( l \in \{1, \ldots, |p|\} \) from (2a) and (2b). Furthermore, it is clear that there exists a mapping between \( (h_{i1}, \ldots, h_{ip}) \) and \( (T_{i1}, \ldots, T_{ip}) \). By simple algebra, we obtain

\[
T_{i1} = \frac{1}{\lambda_i} \log \left( \frac{1 + \frac{h_{i1}}{1 - (h_{i1} + h_{i2} + \cdots + h_{ip})}}{\frac{h_{i1}}{h_{i1} - 1}} \right) ,
\]

\[
T_{il} = \frac{1}{\lambda_i} \log \left( \frac{1 + \frac{h_{il}}{h_{i(l-1)}}}{\frac{h_{il}}{h_{i(l-1)}}} \right) , \quad l = 2, \ldots, |p| ,
\]

Note that

\[
0 < h_{i1} + h_{i2} + \cdots + h_{ip} \leq 1 ,
\]

must hold during the operation, which is always true for our caching policies.

C. Maximizing Aggregate Utility

With the change of variables discussed above, we can reformulate (5) as follows

\[
\max_{T} \sum_{i \in \mathcal{D}} \sum_{l=1}^{|p|} \psi_i |p| - l U_i (\lambda_i, h_{il})
\]
and solving the following optimization problem

$$\text{s.t. } \sum_{i \in \mathcal{D}} h_{il} \leq B_l, \quad l = 1, \ldots, |\mathcal{D}|,$$  \hspace{1cm} (8b)  

$$|\mathcal{D}| \sum_{i \in \mathcal{D}} h_{il} \leq 1, \quad \forall i \in \mathcal{D},$$  \hspace{1cm} (8c)  

$$0 \leq h_{il} \leq 1, \quad \forall i \in \mathcal{D}, \quad l = 1, \ldots, |\mathcal{D}| \quad (8d)$$

where (8b) is the cache capacity constraint and (8c) is due to the variable exchanges under MCDP as discussed above.

**Proposition 1.** Optimization problem defined in (8) under MCDP has a unique global optimum.

### D. Upper Bound (UB) on optimal aggregate utility

Constraint (8c) in (8) is enforced due to variable exchanges under MCDP as discussed above. Here we can define an upper bound on optimal aggregate utility by removing (8c) and solving the following optimization problem

$$\max \sum_{i \in \mathcal{D}} \sum_{l=1}^{|\mathcal{D}|} \psi_{il} |\mathcal{D}| - U_i(\lambda_i h_{il}),$$  

s.t. constraints (8b) and (8d). \hspace{1cm} (9)

Note that the UB optimization problem is now independent of any timer driven caching policy and can be used as a performance benchmark for comparing various caching policies. Furthermore, it is easier to solve UB optimization problem (9) since it involves a smaller number of constraints compared to MCDP based optimization problem (8).

### E. Distributed Algorithm

In Section III-C, we formulated convex utility maximization problems with a fixed cache size. However, system parameters (e.g., cache size and request processes) can change over time, so it is not feasible to solve the optimization offline and implement the optimal strategy. Thus, we need to design distributed algorithms to implement the optimal strategy and adapt to the changes in the presence of limited information.

**Primal Algorithm:** We aim to design an algorithm based on the optimization problem in (8), which is the primal formulation. Denote \( h_i = (h_{i1}, \ldots, h_{i|\mathcal{D}|}) \) and \( h = (h_1, \ldots, h_n) \). We first define the following objective function.

$$Z(h) = \sum_{i \in \mathcal{D}} \sum_{l=1}^{|\mathcal{D}|} \psi_{il} |\mathcal{D}| - U_i(\lambda_i h_{il}) - C_l \left( \sum_{i \in \mathcal{D}} h_{il} - B_l \right)$$  

$$- \sum_{i \in \mathcal{D}} \tilde{C}_i \left( \sum_{l=1}^{|\mathcal{D}|} h_{il} - 1 \right) - \sum_{i \in \mathcal{D}} \sum_{l=1}^{|\mathcal{D}|} \tilde{C}_il(-h_{il}), \hspace{1cm} (10)$$

where \( C_l(\cdot), \tilde{C}_i(\cdot) \) and \( \tilde{C}_il(\cdot) \) are convex and non-decreasing penalty functions denoting the cost for violating constraints (8b) and (8c).

Note that constraint (8c) ensures \( h_{il} \leq 1 \) \( \forall i \in \mathcal{D}, \ l = 1, \ldots, |\mathcal{D}| \), provided \( h_{il} \geq 0 \). One can assume that \( h_{il} \geq 0 \) holds in writing down (10). This would be true, for example, if the utility function is a \( \beta \)-fair utility function with \( \beta > 0 \) (Section 2.5 [17]). For other utility functions, it is challenging to incorporate constraint (8d) since it introduces \( n|\mathcal{D}| \) additional price functions. For all cases evaluated across various system parameters we found \( h_{il} \geq 0 \) to hold true. Hence we ignore constraint (8d) in the primal formulation and define the following objective function

$$Z(h) = \sum_{i \in \mathcal{D}} \sum_{l=1}^{|\mathcal{D}|} \psi_{il} |\mathcal{D}| - U_i(\lambda_i h_{il}) - C_l \left( \sum_{i \in \mathcal{D}} h_{il} - B_l \right)$$  

$$- \sum_{i \in \mathcal{D}} \tilde{C}_i \left( \sum_{l=1}^{|\mathcal{D}|} h_{il} - 1 \right).$$  \hspace{1cm} (11)

It is clear that \( Z(\cdot) \) is strictly concave. Hence, a natural way to obtain the maximal value of (11) is to use the standard gradient ascent algorithm to move the variable \( h_{il} \) for \( i \in \mathcal{D} \) and \( l \in \{1, \ldots, |\mathcal{D}|\} \) in the direction of gradient,

$$\frac{\partial Z(h)}{\partial h_{il}} = \lambda_i \psi_{il} |\mathcal{D}| - U_i'(\lambda_i h_{il}) - C_l' \left( \sum_{j \in \mathcal{D}} h_{jl} - B_l \right) - \tilde{C}_i' \left( \sum_{m=1}^{|\mathcal{D}|} h_{im} - 1 \right),$$  \hspace{1cm} (12)

where \( U_i'(\cdot), C_l'(\cdot), \tilde{C}_i'(\cdot) \) denote partial derivatives w.r.t. \( h_{il} \).

Since \( h_{il} \) indicates the probability that content \( i \) is in cache \( l \), \( \sum_{j \in \mathcal{D}} h_{jl} \) is the expected number of contents currently in cache \( l \), denoted by \( B_{curr,l} \).

Therefore, the primal algorithm for MCDP is given by

$$T_{il}[k] \left\{ \begin{array}{ll} \frac{1}{\lambda_i} \log \left( 1 + \frac{h_{il}[k]}{1 - (h_{il}[k] + h_{il}[k] + \cdots + h_{il}[k])} \right), & l = 1; \\ \frac{1}{\lambda_i} \log \left( 1 + \frac{h_{il}[k]}{1 - (h_{il}[k] + \cdots + h_{il}[k])} \right), & l = 2, \ldots, |\mathcal{D}|, \hspace{1cm} (13a) \end{array} \right.$$

$$h_{il}[k+1] \left\{ \begin{array}{ll} \max \left\{ 0, h_{il}[k] + \zeta_{il} \lambda_i \psi_{il} |\mathcal{D}| - U_i'(\lambda_i h_{il}[k]) \\ - C_l'(B_{curr,l} - B_l) - \tilde{C}_i' \left( \sum_{m=1}^{|\mathcal{D}|} h_{im}[k] - 1 \right) \right\}, \hspace{1cm} (13b) \end{array} \right.$$

where \( \zeta_{il} > 0 \) is the step-size parameter, and \( k \) is the iteration number incremented upon each request arrival.

**Theorem 1.** The primal algorithm (13) is locally asymptotically stable given a sufficiently small step-size parameter \( \zeta_{il} \).

**Proof.** Since \( U_i(\cdot) \) is strictly concave, \( C_l(\cdot) \) and \( \tilde{C}_i(\cdot) \) are convex, then (11) is strictly concave, hence there exists a unique maximizer. Denote it as \( h^* \).

Any differentiable function \( f(x) \) can be linearized around a point \( x^* \) as \( L(x) = f(x^*) + f'(x^*)(x - x^*) \). Denote \( \forall i, l \)

$$f(h_{il}) = h_{il} + \zeta_{il} \left[ \lambda_i \psi_{il} |\mathcal{D}| - U_i'(\lambda_i h_{il}) - C_l' \left( \sum_{j \in \mathcal{D}} h_{jl} - B_l \right) - \tilde{C}_i' \left( \sum_{m=1}^{|\mathcal{D}|} h_{im} - 1 \right) \right], \hspace{1cm} (14)$$
with \( f : \mathbb{R}^+ \rightarrow \mathbb{R} \). We have \( f(h_{il}) = h_{il}^* \). Under linearization,

\[
    h_{il}[k + 1] = h_{il}^* + f'(h_{il}^*)(h_{il}[k] - h_{il}^*).
\]

(15)

Denote \( h_{il}^*[k] = h_{il}[k] - h_{il}^* \) as deviation from \( h_{il}^* \) at \( k^{th} \) iteration. Hence we have

\[
    h_{il}^*[k + 1] = f'(h_{il}^*)h_{il}^*[k] = [f'(h_{il}^*)]^k h_{il}^*[0].
\]

(16)

Thus (15) is locally asymptotically stable if

\[
    |f'(h_{il}^*)| < 1.
\]

Computing \( f'(h_{il}^*) \) from (14) and substituting in (17) yields

\[
    \eta_{il} < \frac{2}{\lambda x \psi(|p| - 1)U''(\lambda x_{il})}
\]

(18)

Note that, since the functions \( C_l, \bar{C}_l \) and \( U_l \) are strictly convex, strictly convex and strictly concave functions respectively, \( C''(x) < 0, \bar{C}''(x) < 0 \) and \( U''(x) < 0 \) \( \forall x \in \mathbb{R}^+ \). Hence the r.h.s of (18) is strictly positive and for a sufficiently small positive step-size parameter \( \eta_{il} \), (18) always holds. Thus the update rule (11) converges to \( h_{il}^* \) as long as \( h_{il}^{(0)} \) is sufficiently close to \( h_{il}^* \) for all \( i \in D \) and \( l = 1, 2, \ldots, |p| \).

\[\Box\]

**Remark 2.** Note that the primal formulation in (13) can be implemented distributively with respect to (w.r.t.) different contents and caches by some amount of book-keeping. For example in (13b), \( \sum_{m=1}^{|p|} h_{im}[k] \) at cache \( l \) can be computed by first storing the value of \( \sum_{m=1}^{|p|} h_{im}[k] \) at the edge cache in previous iteration and updating it during delivery of content \( i \) (from cache \( l \)) to the user.

### F. Model Validation and Insights

We validate our analytical results with simulations for MCDP. We consider a three-node path with cache capacities \( B_l = 10, \ l = 1, 2, 3 \). The total number of unique contents considered in the system is \( n = 100 \). We consider the Zipf popularity distribution with parameter \( \alpha = 0.8 \). W.l.o.g., we consider log based utility function \( U_i(x) = \lambda_i \log(1 + x) \) \( \{18\} \), and discount factor \( \psi = 0.1 \). We assume that requests arrive according to a Poisson process with aggregate request rate \( \Lambda = 1 \).

We first solve optimization problem (3) using a Matlab routine \texttt{fmincon}. From Figure 2(a), we observe that popular contents are assigned higher hit probabilities at cache node 1, i.e. at the edge cache closest to the user as compared to other caches. The optimal hit probabilities assigned to popular contents at other caches are almost negligible. However, the assignment is reversed for moderately popular contents. For non-popular contents, optimal hit probabilities at cache node 1 (closest to origin server) are the highest.

We then implement our primal algorithm given in (13), where we take the following penalty functions (17) \( C_l(x) = \max\{0, x - B_l \log(B_l + x)\} \) and \( \bar{C}_l(x) = \max\{0, x - \log(1 + x)\} \). Figure 2(b) shows that the primal algorithm successfully converges to the optimal solution.

We also compare the performance of MCDP to other policies such as K-LRU (K=3), K-LRU with big cache abstraction: K-LRU(B) and the UB based bound. We plot the relative performance w.r.t. the optimal aggregated utilities of all above policies, normalized to that under MCDP shown in Figure 3. We observe that MCDP significantly outperforms K-LRU and K-LRU(B) for low and moderate values of Zipf parameter.
Furthermore, the performance gap between UB and MCDP increases in Zipf parameter.

Finally, we consider the effect of $\alpha$ and $\psi$ on the performance gap (difference between optimal aggregate utility) between UB and MCDP. We present the simulation results in Figure 4. Note that utility under both UB and MCDP increases in either $\alpha$ or $\psi$ or both. When either $\alpha$ or $\psi$ is small, irrespective of the value of the other, the performance gap is minor or negligible. However, the gap is considerably large when both $\alpha$ and $\psi$ are high. We believe, due to high communication overhead between successive layers of caches, $\psi$ has a small value, thus indicating a minor performance gap between MCDP and UB.

IV. GENERAL EDGE NETWORKS

In Section III, we consider linear edge networks and characterize the optimal TTL policy for content when coupled with MCDP. Inspired by these results, we consider general edge networks in this section.

A. Contents, Servers and Requests

We consider the general edge network described in Section II. Denote by $\mathcal{P}$ the set of all requests, and $\mathcal{P}^i$ the set of requests for content $i$. Suppose a cache in node $v$ serves two requests $(v_1, i_1, p_1)$ and $(v_2, i_2, p_2)$, then there are two cases: (i) non-common requested content, i.e., $i_1 \neq i_2$; and (ii) common requested content, i.e., $i_1 = i_2$.

B. Non-common Requested Content

In this section, we consider the case that each network cache serves requests for different contents from each request $(v, i, p)$ passing through it. Since there is no coupling between different requests $(v, i, p)$, we can directly generalize the results for a particular path $p$ in Section III to a tree network. Hence, given the utility maximization formulation in (8), we can directly formulate the optimization problem for MCDP as

$$\max \sum_{i \in \mathcal{D}} \sum_{p \in \mathcal{P}^i} \sum_{l=1}^{\pi} \psi^{l-1} U_{ip}(\lambda_{ip} h_{il}^{(p)})$$

s.t.

$$\sum_{i \in \mathcal{D}} \sum_{p \in \{1, \ldots, |\pi|\}} h_{il}^{(p)} \leq B_i, \quad \forall l \in V,$$  

$$\sum_{l=1}^{\pi} h_{il}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, p \in \mathcal{P}^i,$$  

$$0 \leq h_{il}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, l \in \{1, \ldots, |\pi|\}, p \in \mathcal{P}^i.$$  

Proposition 2. Since the feasible sets are convex and the objective function is strictly concave and continuous, the optimization problem defined in (19) under MCDP has a unique global optimum.

1) Model Validation and Insights: We consider a three-layer edge network shown in Figure 4 with node set \{1, \ldots, 7\}, which is consistent with the YouTube video delivery system [11], [12]. Nodes 1-4 are edge caches, and node 7 is tertiary cache. There exist four paths $p_1 = \{1, 5, 7\}$, $p_2 = \{2, 5, 7\}$, $p_3 = \{3, 6, 7\}$ and $p_4 = \{4, 6, 7\}$. Each edge cache serves requests for 100 distinct contents, and cache size is $B_i = 10$ for $v \in \{1, \ldots, 7\}$. Assume that content follows a Zipf distribution with parameter $\alpha_1 = 0.2$, $\alpha_2 = 0.4$ and $\alpha_4 = 0.8$. We consider utility function $U_{ip}(x) = \lambda_{ip} \log(1 + x)$, where $\lambda_{ip}$ is the request arrival rate for content $i$ on path $p$, and, requests are described by a Poisson process with $\Lambda_{ip} = 1$ for $p = 1, 2, 3, 4$. The discount factor $\psi = 0.1$.

Figure 5 shows results for path $p_4 = \{4, 6, 7\}$. From Figure 5 (Left), we observe that our algorithm yields the exact optimal and empirical hit probabilities under MCDP. Figure 5 (Right) shows the probability density of the number of content requests in the edge network. As expected, the density is concentrated around their corresponding cache sizes. Similar trends exist for paths $p_1$, $p_2$ and $p_3$, hence are omitted here.

C. Common Requested Contents

Now consider the case where different users share the same content, e.g., there are two requests $(v_1, i, p_1)$ and $(v_2, i, p_2)$. Suppose that cache $l$ is on both paths $p_1$ and $p_2$, where $v_1$ and $v_2$ request the same content $i$. If we cache separate copies on each path, results from the previous section apply. However, maintaining redundant copies in the same cache decreases efficiency. A simple way to deal with that is to only cache one copy of content $i$ at $l$ to serve both requests from $v_1$ and $v_2$. Though this reduces redundancy, it complicates the optimization problem.

In the following, we formulate a utility maximization problem for MCDP with TTL caches, where all users share the same requested contents $\mathcal{D}$.

$$\max \sum_{i \in \mathcal{D}} \sum_{p \in \mathcal{P}^i} \sum_{l=1}^{\pi} \psi^{l-1} U_{ip}(\lambda_{ip} h_{il}^{(p)})$$

The constraint (19b) in problem (19) is on average cache occupancy. However, it can be shown that if $n \to \infty$ and $B_i$ grows in sub-linear manner, the probability of violating the target cache size $B_i$ becomes negligible [19].
Consider two requests $MCDP$ TTL cache policy as discussed in Section III-B.

\[ \sum_{i \in D} (1 - \prod_{p \in [1, \ldots, |p|]} (1 - h^{(p)}_{ij})) \leq B_j, \quad \forall j \in V, \quad (20b) \]

\[ \sum_{j \in [1, \ldots, |p|]} h^{(p)}_{ij} \leq 1, \quad \forall i \in D, p \in P^i, \quad (20c) \]

\[ 0 \leq h^{(p)}_{ij} \leq 1, \quad \forall i \in D, j \in \{1, \ldots, |p|\}, p \in P^i, \quad (20d) \]

where (20b) ensures that only one copy of content $i \in D$ is cached at node $j$ for all paths $p$ that pass through node $j$. This is because the term $1 - \prod_{p : j \in [1, \ldots, |p|]} (1 - h^{(p)}_{ij})$ is the overall hit probability of content $i$ at node $j$ over all paths. (20c) is the cache capacity constraint and (20d) is the constraint from $MCDP$ TTL cache policy as discussed in Section III-B.

**Example 1.** Consider two requests $(v_1, i, p_1)$ and $(v_2, i, p_2)$ with paths $p_1$ and $p_2$ intersecting at $j$. Let the corresponding path perspective hit probability be $h^{(p)}_{ij}$ and $h^{(p)}_{ij}$. Then the term inside outer summation of (20b) is $1 - (1 - h^{(p)}_{ij})(1 - h^{(p)}_{ij})$, i.e., the hit probability of content $i$ in $j$.

**Remark 3.** Note that we assume independence between different requests $(v, i, p)$ in (20), e.g., in Example 1 if the insertion of content $i$ in node $j$ is caused by request $(v_1, i, p_1)$, when request $(v_2, i, p_2)$ comes, it is not counted as a cache hit from its perspective. Our framework still holds if we follow the logical TTL $MCDP$ on a path. However, in that case, the utilities will be larger than the one we consider.

**Proposition 3.** Since the feasible sets are non-convex, under $MCDP$ is a non-convex optimization problem.

In the following, we develop an optimization framework that handles the non-convex issue in this optimization problem and provides a distributed solution. To this end, we first introduce the Lagrangian function

\[
L(h, \nu, \mu) = \sum_{i \in D} \sum_{p \in P^i} \sum_{l=1}^{|p|} \psi^{|p|-l} U_{ip}(\lambda_{ip} h^{(p)}_{il})
- \sum_{j \in V} \nu_j \left( \sum_{i \in D} \left[ 1 - \prod_{p : j \in [1, \ldots, |p|]} (1 - h^{(p)}_{ij}) \right] - B_j \right)
- \sum_{i \in D} \sum_{p \in P^i} \mu_{ip} \left( \sum_{j \in [1, \ldots, |p|]} h^{(p)}_{ij} - 1 \right),
\]

where the Lagrangian multipliers (price vector and price matrix) are $\nu = (\nu_j)_{j \in V}$ and $\mu = (\mu_{ip})_{(i,p) \in D \times P}$. Constraint (20d) is ignored in the Lagrangian function due to the same reason stated for the primal formulation in Section III-E.

The dual function can be defined as

\[ d(\nu, \mu) = \sup_h L(h, \nu, \mu), \quad (22) \]

and the dual problem is given as

\[ \min_{\nu, \mu} d(\nu, \mu) = L^*(\nu, \mu, \nu, \mu), \quad \text{s.t. } \nu, \mu \geq 0, \quad (23) \]

where the constraint is defined pointwise for $\nu, \mu,$ and $h^*(\nu, \mu)$ is a function that maximizes the Lagrangian function for given $(\nu, \mu)$, i.e.,

\[ h^*(\nu, \mu) = \arg \max_h L(h, \nu, \mu). \quad (24) \]

The dual function $d(\nu, \mu)$ is always convex in $(\nu, \mu)$ regardless of the convexity of the optimization problem (20). Therefore, it is always possible to iteratively solve the dual problem using

\[ \nu_l[k + 1] = \nu_l[k] - \gamma \frac{\partial L(\nu, \mu)}{\partial \nu_l}, \]

\[ \mu_{ip}[k + 1] = \mu_{ip}[k] - \eta_{ip} \frac{\partial L(\nu, \mu)}{\partial \mu_{ip}}, \quad (25) \]

where $\gamma$ and $\eta_{ip}$ are the step sizes, and $\frac{\partial L(\nu, \mu)}{\partial \nu_l}$ and $\frac{\partial L(\nu, \mu)}{\partial \mu_{ip}}$ are the partial derivative of $L(\nu, \mu)$ w.r.t. $\nu_l$ and $\mu_{ip}$, respectively, satisfying

\[ \frac{\partial L(\nu, \mu)}{\partial \nu_l} = - \left( \sum_{i \in D} \sum_{p : j \in [1, \ldots, |p|]} (1 - h^{(p)}_{ij}) - B_j \right), \]

\[ \frac{\partial L(\nu, \mu)}{\partial \mu_{ip}} = - \left( \sum_{j \in [1, \ldots, |p|]} h^{(p)}_{ij} - 1 \right). \quad (26) \]

Sufficient and necessary conditions for the uniqueness of $\nu, \mu$ are given in (21). The convergence of primal-dual algorithm consisting of (24) and (25) is guaranteed if the original optimization problem is convex. However, our problem is not convex. Nevertheless, we next show that the duality gap is zero, hence (24) and (25) converge to the globally optimal solution. To begin with, we introduce the following results

**Theorem 2.** (Sufficient Condition). If the price based function $h^*(\nu, \mu)$ is continuous at one or more of the optimal lagrange multiplier vectors $\nu^*$ and $\mu^*$, then the iterative algorithm consisting of (24) and (25) converges to the globally optimal solution.

**Theorem 3.** (Necessary Condition). If at least one constraint of (20) is active at the optimal solution, the condition in Theorem 2 is also a necessary condition.

Hence, if we can show the continuity of $h^*(\nu, \mu)$ and that constraints (20) are active, then given Theorems 2 and 3 the duality gap is zero, i.e., (24) and (25) converge to the globally optimal solution.

Take the derivative of $L(h, \nu, \mu)$ w.r.t. $h^{(p)}_{il}$ for $i \in D$, $l \in \{1, \ldots, |p|\}$ and $p \in P^i$, we have

\[ \frac{\partial L(h, \nu, \mu)}{\partial h^{(p)}_{il}} = \psi^{|p|-l} \lambda_{ip} U_{ip}(\lambda_{ip} h^{(p)}_{il}) - \mu_{ip} - \nu_l \left( \prod_{q \neq p : j \in [1, \ldots, |p|]} (1 - h^{(q)}_{ij}) \right). \quad (27) \]

Setting (27) equal to zero, we obtain

\[ U_{ip}'(\lambda_{ip} h^{(p)}_{il}) = \frac{1}{\psi^{|p|-l} \lambda_{ip}} \nu_l \left( \prod_{q \neq p : j \in [1, \ldots, |p|]} (1 - h^{(q)}_{ij}) \right) + \mu_{ip}. \quad (28) \]
Consider the utility function \( U_{ip}(\lambda_{ip}h_{it}^{(p)}) = w_{ip}\log(1 + \lambda_{ip}h_{it}^{(p)}) \), then \( U'_{ip}(\lambda_{ip}h_{it}^{(p)}) = w_{ip}/(1 + \lambda_{ip}h_{it}^{(p)}) \). Hence, from (28), we have
\[
\begin{align*}
    h_{it}^{(p)} &= \frac{w_{ip}v_i^{p|l| - l}}{\nu_i \left( \prod_{j \in \{1, \ldots, |q|\} : q \neq p} (1 - h_{ij}^{(q)}) \right) + \mu_{ip}} - \frac{1}{\lambda_{ip}}, \\
    (29)
\end{align*}
\]

Lemma 1. Constraints (20b) and (20c) cannot be both non-active, i.e., at least one of them is active.

Proof. We prove this lemma by contradiction. Suppose both constraints (20b) and (20c) are non-active, i.e., \( \nu = (0) \) and \( \mu = (0) \). Then the optimization problem (19) achieves its maximum when \( h_{it}^{(p)} = 1 \) for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \). If so, then the left hand size of (20b) equals \( \mathcal{B}_0(\lambda_0) \) which is much greater than \( \mathcal{B}_0 \) for \( l \in V \), which is a contradiction. Hence, constraints (20b) and (20c) cannot be both non-active.

From Lemma 1, we know that the feasible region for the Lagrangian multipliers satisfies \( \mathcal{R} = \{ \nu_i \geq 0, \mu_{ip} \geq 0, \nu_i + \mu_{ip} \neq 0, i \in D, l \in \{1, \ldots, |p|\}, p \in \mathcal{P}^i \} \).

Theorem 4. The hit probability \( h_{it}^{(p)} \) given in (29) is continuous in \( \nu_i \) and \( \mu_{ip} \) for all \( i \in D, l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \) in the feasible region \( \mathcal{R} \).

Proof. From Lemma 1, we know at least one of \( \nu_i \) and \( \mu_{ip} \) is non-zero, for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \). Hence there are three cases, (i) \( \nu_i \neq 0 \) and \( \mu_{ip} = 0 \); (ii) \( \nu_i = 0 \) and \( \mu_{ip} \neq 0 \); and (iii) \( \nu_i \neq 0 \) and \( \mu_{ip} \neq 0 \).

For case (i), we have
\[
\begin{align*}
    h_{it}^{(p)} &= \frac{w_{ip}v_i^{p|l| - l}}{\nu_i \left( \prod_{j \in \{1, \ldots, |q|\} : q \neq p} (1 - h_{ij}^{(q)}) \right) + \mu_{ip}} - \frac{1}{\lambda_{ip}}, \\
    (30)
\end{align*}
\]
which is clearly continuous in \( \nu_i \), for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \).

Similarly for case (ii), we have
\[
\begin{align*}
    h_{it}^{(p)} &= \frac{w_{ip}v_i^{p|l| - l}}{\mu_{ip}} - \frac{1}{\lambda_{ip}}, \\
    (31)
\end{align*}
\]
which is also clearly continuous in \( \mu_{ip} \), for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \).

For case (iii), from (29), it is obvious that \( h_{it}^{(p)} \) is continuous in \( \nu_i \) and \( \mu_{ip} \) for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \).

Therefore, we know that \( h_{it}^{(p)} \) is continuous in \( \nu_i \) and \( \mu_{ip} \) for all \( i \in D \), \( l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \).

Remark 4. Note that similar arguments (by using Lemma 1) hold true for various other choice of utility functions such as: \( \beta \) fair utility functions (Section 2 [17]). Therefore, the primal-dual algorithm consisting of (24) and (25) converges to the globally optimal solution for a wide range of utility functions.

Algorithm 1 summarizes the details of this algorithm.

### Algorithm 1 Primal-Dual Algorithm

**Input:** \( \forall \nu_0, \mu_0 \in \mathcal{R} \) and \( h_0 

**Output:** The optimal hit probabilities \( h 

**Step 0:** \( t = 0 \), \( \nu[t] \leftarrow \nu_0 \), \( \mu[t] \leftarrow \mu_0 \), \( h[t] \leftarrow h_0 

**Step t \geq 1**

**while** Equation (25) \( \neq 0 \) **do**

1. **First**, compute \( h_{it}[t+1] \) for \( i \in D, l \in \{1, \ldots, |p|\} \) and \( p \in \mathcal{P}^i \) through (29).

2. **Second**, update \( \nu_i[t+1] \) and \( \mu_{ip}[t+1] \) through (25) given \( h[t+1] \), \( \nu[t] \) and \( \mu[t] \) for \( l \in V, i \in D \) and \( p \in \mathcal{P}^i \).

1) **Model Validation and Insights:** We evaluate the performance of Algorithm 1 on a three-layer edge network shown in Figure 1. We assume that there are totally 100 unique contents in the system requested from four paths. The cache size is given as \( B_v = 10 \) for \( v = 1, \ldots, 7 \). We consider the utility function \( U_{ip}(x) = \lambda_{ip}\log(1 + x) \), and the popularity distribution over these contents is Zipf with parameter 0.8. W.l.o.g., the aggregate request arrival rate is one. The discount factor \( \psi = 0.1 \).

We solve the optimization problem in (20) using a Matlab routine fmincon. Then we implement our primal-dual algorithm given in Algorithm 1. The result for aggregate optimal utility is presented in Figure 6. It is clear that the primal-dual algorithm successfully converges to the optimal solution.

Similar to Section III-F, we compare MCDP to K-LRU, K-LRU(B) and UB. Figure 7 compares the performance of different eviction policies to our MCDP policy. We plot the relative performance w.r.t. the optimal aggregated utilities of all above policies, normalized to that under MCDP. We again observe a huge gain of MCDP w.r.t. K-LRU and K-LRU(B) across all values of discount factor. However, the performance gap between MCDP and UB increases with an increase in the value of discount factor.

Finally, we consider how cache capacity affects the overall objective for both MCDP and UB. Define \( \eta = B_v/n, \forall v \in V \) as the ratio of cache size to total number of contents. For our numerical studies, we take \( \eta = \eta, \forall v \in V \). From Figure 8, we observe that the aggregate optimal utility increases in \( \eta \). Note that as \( \eta \) increases, the aggregate optimal utility gradually converges to a value, and then becomes insensitive to \( \eta \). For a range of simulations performed with different system parameters, we find \( \eta \in (0.2, 0.3) \) to be high enough to obtain maximum achievable aggregate utility. This provides design guidelines on how to set cache size for real systems [11], [15], [23], [24]. Furthermore, the performance gap between UB and MCDP first increases in \( \eta \) and later becomes insensitive to \( \eta \).

V. APPLICATIONS TO GENERAL GRAPHS

In this section, we consider a direct application of our framework to content distributions in CDNs, ICNs etc. The problem we consider so far is directly motivated by and naturally captures many important realistic networking applications. These include the Web [5], the domain name system (DNS) [25], [26], content distribution networks (CDNs) [27], [28], information and content-centric networks (ICNs/CCNs) [29], [30].
named data networks (NDNs) [29], etc. For example, in modern CDNs with hierarchical topologies, requests for content can be served by intermediate caches placed at edge server that acts as a designated source in the domain. Similarly, in ICNs/NDNs, named data are stored at designated servers. Requests for named data are routed to the server, which can be stored at intermediate routers to serve future requests. Both settings directly map to the problem we study here (Section IV-C). In particular, we consider content distribution in this section. These present hard problems: highly diverse traffic with different content types, such as videos, music and images, require CDNs to cache and deliver content using a shared distributed cache server infrastructure so as to obtain economic benefits. Our timer-based model with simple cache capacity constraint enable us to provide optimal and distributed algorithms for these applications.

Here we consider a general network topology with overlapping paths and common contents requested along different paths. Similar to [20] and Algorithm [1], a non-convex optimization problem and a primal-dual algorithm can be formulated and designed. Due to space constraints, we omit the details and only show the performance of this general network.

We consider two network topologies: Grid and lollipop. Grid is a two-dimensional square grid while a (a,b) lollipop network is a complete graph of size $a$, connected to a path graph of length $b$. Denote the network as $G = (V,E)$. For grid, we consider $|V| = 16$, while we consider a (3,4) lollipop topology with $|V| = 7$ and clique size 3. The library contain $|D| = 100$ unique contents. Each node has access to a subset of contents in the library. We assign a weight to each edge in $E$, selected uniformly from the interval $[1,20]$. Next, we generate a set of requests in $G$ as described in [30]. To ensure that paths overlap, we randomly select a subset $V \subset V$ nodes to generate requests. Each node in $V$ can generate requests for contents in $D$ following a Zipf distribution with parameter $\alpha = 0.8$. Requests are then routed over the shortest path between the requesting node in $V$ and the node in $V$ that caches the content. Again, we assume that the aggregate request rate at each node in $V$ is one and the discount factor to be $\psi = 0.1$.

We evaluate the performance of MCDP over the graphs across various Zipf parameter in Table 1. It is clear that for both network topologies, aggregate utility obtained from our TTL-based framework with MCDP policy is higher for higher Zipf parameter as compared to lower Zipf parameter. With increase in Zipf parameter, the difference between request rates of popular and less popular contents increases. The aggregate request rate over all contents is the same in both cases. Thus popular contents get longer fraction of rates which in turn yields higher aggregate utility. However, the performance gap between UB and MCDP is around one tenth in both cases and is not affected much by the Zipf parameter.

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|c|}
\hline
Topology & $\alpha$ & MCDP & UB & \% Gap \\
\hline
Grid & 0.8 & 0.09235 & 0.1043 & 11.50 \\
Grid & 1.2 & 0.3611 & 0.4016 & 10.08 \\
Lollipop & 0.8 & 0.0908 & 0.1002 & 9.38 \\
Lollipop & 1.2 & 0.3625 & 0.4024 & 9.91 \\
\hline
\end{tabular}
\caption{Optimal aggregate utilities under various network topologies.}
\end{table}

VI. MINIMIZING OVERALL COSTS

In Section III, we focus on maximizing the sum of utilities across all contents over the edge network, which captures user satisfaction. However, communication costs for content transfers across the network are also critical in many network applications. This cost includes (i) search cost for finding the requested content in the network; (ii) fetch cost to serve the content to the user; and (iii) transfer cost for cache inner management due to a cache hit or miss. Below we derive expressions for evaluating these costs for MCDP policy. We relegate the cost analysis for MCD policy to Appendix D.

A. Search and Fetch Cost

A request is sent along a path until it hits a cache that stores the requested content. We define search cost (fetch cost) as the cost of finding (serving) the requested content in the cache network (to the user). Consider cost as a function $c_s(\cdot)$ ($c_f(\cdot)$) of the hit probabilities. Then the expected search cost across the network is given as

$$S_{\text{MCDP}} = \sum_{i \in D} \lambda_i c_s \left( \sum_{l=0}^{|p|} (|p| - l + 1) h_{si} \right).$$

(32)

Fetch cost has a similar expression with $c_f(\cdot)$ replacing $c_s(\cdot)$.
B. Transfer Cost

Under TTL, upon a cache hit, the content either transfers to a higher index cache or stays in the current one, and upon a cache miss, the content transfers to a lower index cache (MCDP). We define transfer cost as the cost due to cache management upon a cache hit or miss. Consider the cost as a function \( c_m(\cdot) \) of the hit probabilities.

Note that under MCDP, there is a transfer upon a content request or a timer expiry except two cases: (i) content \( i \) is in cache 1 and a timer expiry occurs, which occurs with probability \( \pi_{i1}e^{-\lambda_i T_{i1}} \); and (ii) content \( i \) is in cache \( |p| \) and a cache hit (request) occurs, which occurs with probability \( \pi_{i|p|}(1 - e^{-\lambda_i T_{i|p|}}) \). Then the transfer cost for content \( i \) at steady state is

\[
M_{\text{MCDP}}^i = \lim_{n \to \infty} M_{\text{MCDP}}^i = \lim_{n \to \infty} \frac{1}{n} \sum_{j=1}^{n} (t_j^i - t_{j-1}^i) = \frac{1 - \pi_{i1}e^{-\lambda_i T_{i1}} - \pi_{i|p|}(1 - e^{-\lambda_i T_{i|p|}})}{\sum_{l=0}^{|p|} \pi_{il}E[t_j^i - t_{j-1}^i | X_j^i = l]}
\]

(33)

where \( M_{\text{MCDP}}^{i,j} \) means there is a transfer cost for content \( i \) at the \( j \)-th request or timer expiry, \( E[t_j^i - t_{j-1}^i | X_j^i = l] = \frac{1 - e^{-\lambda_i T_{il}}}{\lambda_i} \) is the average time content \( i \) spends in cache \( l \).

Therefore, the transfer cost for MCDP is

\[
M_{\text{MCDP}} = \sum_{i \in D} \lambda_i M_{\text{MCDP}}^i
\]

\[
= \sum_{i \in D} \lambda_i \left( \frac{1 - \pi_{i1}e^{-\lambda_i T_{i1}} - \pi_{i|p|}(1 - e^{-\lambda_i T_{i|p|}})}{\sum_{l=0}^{|p|} \pi_{il}E[t_j^i - t_{j-1}^i | X_j^i = l]} \right) + \lambda_i (h_1 - h_{i|p|})
\]

(34)

where \((\pi_{i0}, \cdots, \pi_{i|p|})\) for \( i \in D \) is the stationary distribution for the DTMC \( \{X_k^i\}_{k \geq 0} \) defined in Section III-A.

Remark 5. The expected transfer cost \( M_{\text{MCDP}} \) (34) is a function of the timer values. Unlike the problem of maximizing sum of utilities, it is difficult to express \( M_{\text{MCDP}} \) as a function of hit probabilities.

C. Variability Improves Performance

We study the impact of \( \alpha \) on performance for a three node path under the same simulation setting as in Section III-F. We solve (3) for various values of \( \alpha \) and obtain corresponding optimal timers and hit probabilities. We then compute costs associated with the optimal policy using (32) and (34), with linear function \( c_f(c_e) \) with weight 1 (0.1), and plot them in Figure 9. Note the value of search cost is much smaller than fetch cost, which is consistent with practice, hence we ignore its curve in Figure 9. As \( \alpha \) increases, popular contents get longer fraction of request rates under Zipf distributions. Furthermore, as \( \alpha \) increases, the popular contents are cached closer to the user under an optimal policy (See Section III-F), hence, the overall search and fetch cost decreases. However, the transfer cost that constitutes a major proportion of the total cost does not change too much as \( \alpha \) increases. Therefore, the total cost associated with the optimal policy decreases in \( \alpha \).

D. Optimization

Our goal is to determine optimal timer values at each cache on one path in the edge network so that the total costs are minimized. To that end, we formulate the following optimization problem for MCDP

\[
\min \quad S_{\text{MCDP}} + F_{\text{MCDP}} + M_{\text{MCDP}}
\]

Constraints in (5).

(35a) (35b)

Remark 6. As discussed in Remark 5, we cannot express transfer cost of MCDP (34) in terms of hit probabilities, hence, we are not able to transform the optimization problem (35) for MCDP into a convex one through a change of variables as we did in Section III-C. Solving the non-convex optimization problem (35) is a subject of future work.

VII. RELATED WORK

There is a rich literature on the design, modeling and analysis of cache networks, including TTL caches [6], [9], [14], [31], optimal caching [30], [32] and routing policies [33]. In particular, Rodriguez et al. [14] analyzed the advantage of pushing content upstream, Berger et al. [9] characterized the exactness of TTL policy in a hierarchical topology. A unified approach to study and compare different caching policies is given in [15] and an optimal placement problem under a heavy-tailed demand has been explored in [34].

Dehghan et al. [35] as well as Abedini and Shakkottai [50] studied joint routing and content placement with a focus on a bipartite, single-hop setting. Both showed that minimizing single-hop routing cost can be reduced to solving a linear program. Ioannidis and Yeh [33] studied the same problem under a more general setting for arbitrary topologies.

An adaptive caching policy for a cache network was proposed in [30], where each node makes a decision on which
item to cache and evict. An integer programming problem was formulated by characterizing the content transfer costs. Both centralized and complex distributed algorithms were designed with performance guarantees. This work complements our work, as we consider TTL cache and control the optimal cache parameters through timers to maximize the sum of utilities over all contents across the network. However, [30] proposed only approximate algorithms while our timer-based models enable us to design optimal solutions since content occupancy can be modeled as a real variable (e.g., a probability).

Closer to our work, a utility maximization problem for a single cache was considered under IRM [19], [37] and stationary requests [38], while [34] maximized the hit probabilities under heavy-tailed demands over a single cache. None of these approaches generalizes to edge networks, which leads to non-convexity in its full generality, for arbitrary network topologies, overlapping paths and request arrival rates, is one of our technical contributions.

VIII. CONCLUSION

We constructed optimal timer-based TTL policies for content placement in edge networks through a unified optimization approach. We formulated a general utility maximization framework, which is non-convex in general. We identified the non-convexity issue and proposed efficient distributed algorithm to solve it. We proved that the distributed algorithms converge to the globally optimal solutions. We showed the efficiency of these algorithms through numerical studies. An analysis of the MCDP algorithm with reset-TTL timers would remain our future work.

APPENDIX

A. Stationary Behaviors of MCDP

[16] considered a caching policy LRU(m). Though the policy differ from MCDP, the stationary analysis is similar. We present our result here for completeness, which will be used subsequently in the paper.

Under IRM model, the request for content \( i \) arrives according a Poisson process with rate \( \lambda_i \). As discussed earlier, for TTL caches, content \( i \) spends a determinstic time in a cache if it is not requested, which is independent of all other contents. We denote the timer as \( T_{il} \) for content \( i \) in cache \( l \) on the path \( p \), where \( l \in \{1, \cdots, |p|\} \).

Denote \( t_{ik}^l \) as the \( k \)-th time that content \( i \) is either requested or moved from one cache to another. For simplicity, we assume that content is in cache 0 (i.e., server) if it is not in the cache network. Then we can define a discrete time Markov chain (DTMC) \( \{X_k^l\}_{k \geq 0} \) with \(|p| + 1 \) states, where \( X_k^l \) is the cache index that content \( i \) is in at time \( t_{ik}^l \). Since the event that the time between two requests for content \( i \) exceeds \( T_{il} \) happens with probability \( e^{-\lambda_i T_{il}} \), then the transition matrix of \( \{X_k^l\}_{k \geq 0} \) is given as

\[
P_{i}^\text{MCDP} = \begin{bmatrix}
0 & e^{-\lambda_i T_{i1}} & 0 & \cdots & 0 \\
e^{-\lambda_i T_{i(1)}(p-1)} & 1 - e^{-\lambda_i T_{i1}} & e^{-\lambda_i T_{i1}} & \cdots & e^{-\lambda_i T_{i(p-1)}} \\
0 & e^{-\lambda_i T_{i(1)}(p-1)} & 1 - e^{-\lambda_i T_{i1}} & \cdots & e^{-\lambda_i T_{i(p-1)}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & e^{-\lambda_i T_{i(1)}(p-1)} & e^{-\lambda_i T_{i1}} & \cdots & 1 - e^{-\lambda_i T_{i(p-1)}} \\
e^{-\lambda_i T_{i1}} & e^{-\lambda_i T_{i1}} & e^{-\lambda_i T_{i1}} & \cdots & e^{-\lambda_i T_{i1}} \\
\end{bmatrix}.
\]

(36)

Let \( (\pi_{i0}, \cdots, \pi_{i|p|}) \) be the stationary distribution for \( P_{i}^\text{MCDP} \), we have

\[
\pi_{i0} = \frac{1}{1 + \sum_{j=1}^{|p|} e^{\lambda_j T_{ij}} \prod_{s=1}^{j-1} (e^{\lambda_s T_{is}} - 1)},
\]

(37a)

\[
\pi_{i1} = \pi_{i0} e^{\lambda_i T_{ii}},
\]

(37b)

\[
\pi_{il} = \pi_{i0} e^{\lambda_i T_{il}} \prod_{s=1}^{l-1} (e^{\lambda_s T_{is}} - 1), \quad l = 2, \cdots, |p|.
\]

(37c)

Then the average time that content \( i \) spends in cache \( l \in \{1, \cdots, |p|\} \) can be computed as

\[
\mathbb{E}[t_{ik+1}^l - t_{ik}^l | X_k^l = l] = \int_0^{T_{il}} (1 - [1 - e^{-\lambda_i t}]) dt = \frac{1 - e^{-\lambda_i T_{il}}}{\lambda_i},
\]

(38)

and \( \mathbb{E}[t_{ik+1}^l - t_{ik}^l | X_k^l = 0] = \frac{1}{\lambda_i} \).

Given (37) and (38), the timer-average probability that content \( i \) is in cache \( l \in \{1, \cdots, |p|\} \) is

\[
h_{i1} = \frac{e^{\lambda_i T_{i1}} - 1}{1 + \sum_{j=1}^{|p|} (e^{\lambda_i T_{ij}} - 1) \cdots (e^{\lambda_j T_{ij}} - 1)},
\]

\[
h_{il} = h_{i(l-1)} (e^{\lambda_i T_{il}} - 1), \quad l = 2, \cdots, |p|,
\]

where \( h_{il} \) is also the hit probability for content \( i \) at cache \( l \).

B. Stationary Behaviors of Move Copy Down (MCD) policy

MCD behaves the same as MCDP upon a cache hit, i.e. content \( i \) is moved to cache \( l + 1 \) preceding cache \( l \) in which \( i \) is found, and the timer at cache \( l + 1 \) is set to \( T_{(l+1)} \). However, content \( i \) is discarded once the timer expires.

We define a DTMC \( \{Y_k^l\}_{k \geq 0} \) by observing the system at the time that content \( i \) is requested. Similarly, if content \( i \) is not in the cache network, then it is in cache 0, thus we still have \(|p| + 1 \) states. If \( Y_k^l = l \), then the next request for content \( i \) comes within time \( T_{il} \) with probability \( 1 - e^{-\lambda_i T_{il}} \), thus we have \( Y_{k+1}^l = l + 1 \), otherwise \( Y_{k+1}^l = 0 \) due to the MCD policy. Therefore, the transition matrix of \( \{Y_k^l\}_{k \geq 0} \) is given as

\[
P_{i}^\text{MCD} = \begin{bmatrix}
0 & e^{-\lambda_i T_{i1}} & 1 - e^{-\lambda_i T_{i1}} & \cdots & 0 \\
e^{-\lambda_i T_{i1}} & 1 - e^{-\lambda_i T_{i1}} & e^{-\lambda_i T_{i1}} & \cdots & e^{-\lambda_i T_{i(p-1)}} \\
0 & e^{-\lambda_i T_{i1}} & 1 - e^{-\lambda_i T_{i1}} & \cdots & e^{-\lambda_i T_{i(p-1)}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & e^{-\lambda_i T_{i1}} & e^{-\lambda_i T_{i1}} & \cdots & 1 - e^{-\lambda_i T_{i(p-1)}} \\
\end{bmatrix}.
\]

(39)

Let \( (\bar{\pi}_{i0}, \cdots, \bar{\pi}_{i|p|}) \) be the stationary distribution for \( P_{i}^\text{MCD} \), then we have

\[
\bar{\pi}_{i0} = \frac{1}{1 + \sum_{j=1}^{|p|} \prod_{s=1}^{j-1} (1 - e^{-\lambda_j T_{ij}}) + e^{\lambda_i T_{i1}} \prod_{j=1}^{|p|} (1 - e^{-\lambda_j T_{ij}})},
\]

(40a)
\[
\tilde{\pi}_i[l] = \tilde{\pi}_i[0] \prod_{j=1}^{l} (1 - e^{-\lambda_j T_{ij}}), \quad l = 1, \cdots, |p| - 1, \tag{40b}
\]
\[
\tilde{\pi}_i[|p|] = e^{\lambda_1 T_{i1} \tilde{\pi}_i[0] \prod_{j=1}^{|p|-1} (1 - e^{-\lambda_j T_{ij}}). \tag{40c}
\]

By PASTA property [39], we immediately have the that content \( i \) is in cache \( l \in \{1, \cdots, |p|\} \) given as
\[
h_{il} = \tilde{\pi}_i[l], \quad l = 0, 1, \cdots, |p|,
\]
where \( \tilde{\pi}_i \) are given in [40].

C. Optimization Problem for MCD

From [40], we simply check that there exists a mapping between \( (h_{i1}, \cdots, h_{i|p|}) \) and \( (T_{i1}, \cdots, T_{i|p|}) \). By simple algebra, we can obtain
\[
T_{i1} = -\frac{1}{\lambda_i} \log \left( 1 - \frac{h_{i1}}{1 - (h_{i1} + h_{i2} + \cdots + h_{i|p|})} \right), \tag{41a}
\]
\[
T_{il} = -\frac{1}{\lambda_i} \log \left( 1 - \frac{h_{il}}{h_{il-1}} \right), \quad l = 2, \cdots, |p| - 1, \tag{41b}
\]
\[
T_{i|p|} = \frac{1}{\lambda_i} \log \left( 1 + \frac{h_{i|p|}}{h_{i|p|}-1} \right). \tag{41c}
\]

Since \( T_{il} \geq 0 \), we have \( h_{i(|p|-1)} \leq h_{i(|p|-2)} \leq \cdots \leq h_{i1} \leq h_{i0} \). \( \tag{42} \)

and
\[
h_{i1} + h_{i2} + \cdots + h_{i|p|} \leq 1, \tag{43}
\]

must hold during the operation.

1) Non-common Content Requests under General Edge Networks: Similarly, we can formulate a utility maximization optimization problem for MCD under general cache network.

G-N-U-MCD: \[ \max \sum_{i \in \mathcal{D}} \sum_{l \in \mathcal{P}} \sum_{p \in \mathcal{P}^i} \mathcal{C}_s \left( \sum_{l=0}^{|p|} ((|p| - l + 1) h_{il}) \right). \tag{44a} \]

s.t. \[ \sum_{i \in \mathcal{D}} \sum_{l \in \mathcal{P}} h_{il} \leq B_i, \quad \forall p \in \mathcal{P}, \tag{44b} \]

\[ \sum_{l=1}^{|p|} h_{il} \leq 1, \quad \forall i \in \mathcal{D}, \quad \forall p \in \mathcal{P}^i, \tag{44c} \]

\[ h_{il} \leq h_{il} \leq h_{il} \leq h_{il}, \quad \forall p \in \mathcal{P}^i, \tag{44d} \]

\[ 0 \leq h_{il} \leq 1, \quad \forall i \in \mathcal{D}, \quad \forall p \in \mathcal{P}^i. \tag{44e} \]

Proposition 4. Since the feasible sets are convex and the objective function is strictly concave and continuous, the optimization problem defined in (43) under MCD has a unique global optimum.

2) Common Content Requests under General Edge Networks: Similarly, we can formulate the following optimization problem for MCD with TTL caches.

G-U-MCD:
\[ \max \sum_{i \in \mathcal{D}} \sum_{l \in \mathcal{P}} \sum_{p \in \mathcal{P}^i} \mathcal{C}_s \left( \sum_{l=0}^{|p|} ((|p| - l + 1) h_{il}) \right). \tag{45a} \]

s.t. \[ \sum_{i \in \mathcal{D}} \sum_{l \in \mathcal{P}} h_{il} \leq B_i, \quad \forall p \in \mathcal{P}, \tag{45b} \]

\[ \sum_{j \in \{1, \cdots, |p|\}} h_{ij} \leq 1, \quad \forall i \in \mathcal{D}, \quad \forall p \in \mathcal{P}^i, \tag{45c} \]

\[ h_{i(|p|-1)} \leq h_{i(|p|)} \leq h_{i(|p|-1)}, \quad \forall p \in \mathcal{P}^i, \tag{45d} \]

\[ 0 \leq h_{il} \leq 1, \quad \forall i \in \mathcal{D}, \quad \forall p \in \mathcal{P}^i. \tag{45e} \]

Proposition 5. Since the feasible sets are non-convex, the optimization problem defined in (45) under MCD is a non-convex optimization problem.

D. Minimizing Overall Costs for MCD

In the following, we first characterize various costs for MCD. Then we formulate a minimization problem to characterize the optimal TTL policy for content placement in a path.

1) Search Cost: Requests from user are sent along a path until it hits a cache that stores the requested content. We define the search cost as the cost for finding the requested content in the cache network. Consider the cost as a function \( c_s(\cdot) \) of the hit probabilities. Then the expected search cost across the network is given as
\[ S_{MCD} = \sum_{i \in \mathcal{D}} \lambda_i \mathcal{C}_s \left( \sum_{l=0}^{|p|} ((|p| - l + 1) h_{il}) \right). \tag{46} \]

2) Fetch Cost: Upon a cache hit, the requested content will be sent to the user along the reverse direction of the path. We define the fetch cost as the cost of fetching the content to serve the user who sent the request. Consider the cost as a function \( c_f(\cdot) \) of the hit probabilities. Then the expected fetching cost across the network is given as
\[ F_{MCD} = \sum_{i \in \mathcal{D}} \lambda_i \mathcal{C}_f \left( \sum_{l=0}^{|p|} ((|p| - l + 1) h_{il}) \right). \tag{47} \]

3) Transfer Cost: We define the transfer cost as the cost due to caching management upon a cache hit or miss. Consider the cost as a function \( c_m(\cdot) \) of the hit probabilities. Under MCD, since the content is discarded from the network once its timer expires, the transfer cost is caused by a cache hit. To that end, the requested content either moves to a higher index cache if it was in cache \( l \in \{1, \cdots, |p| - 1\} \) or stays in the same cache if it was in cache \( |p| \). Then the expected transfer cost across the network for MCD is given as
\[ M_{MCD} = \sum_{i \in \mathcal{D}} \lambda_i \mathcal{C}_m \left( 1 - h_{il} \right). \tag{48} \]


4) Total Costs: Given the search cost, fetch cost and transfer cost, the total cost for MCD can be defined as
\[
SF_{M_{\text{MCD}}} = S_{M_{\text{MCD}}} + F_{M_{\text{MCD}}} + M_{M_{\text{MCD}}},
\]
where the corresponding costs are given in (46), (47), (48).

REFERENCES

[1] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi, “Orchestrating Massively Distributed CDNs,” in Proc. of ACM CoNEXT, 2012, pp. 133–144.

[2] V. Cisco, “Cisco Visual Networking Index: Forecast and Methodology 2014–2019 White Paper,” Cisco, Tech. Rep., 2015.

[3] M. Feldman and I. Chuang, “Service Differentiation in Web Caching and Content Distribution,” in Proc. of CCN, 2002.

[4] R. Fagin, “Asymptotic Miss Ratios over Independent References,” Journal of Computer and System Sciences, vol. 14, no. 2, pp. 222–250, 1977.

[5] H. Che, Y. Tung, and Z. Wang, “Hierarchical Web Caching Systems: Modeling, Design and Experimental Results,” IEEE Journal on Selected Areas in Communications, vol. 20, no. 7, pp. 1305–1314, 2002.

[6] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, “Performance Evaluation of Hierarchical TTL-based Cache Networks,” Computer Networks, vol. 65, pp. 212–231, 2014.

[7] E. G. Coffman and P. J. Denning, Operating Systems Theory. Prentice-Hall Englewood Cliffs, NJ, 1973, vol. 973.

[8] E. J. Rosensweig, J. Kurose, and D. Towsley, “Approximate Models for General Cache Areas,” in Proc. of IEEE INFOCOM, 2010.

[9] D. S. Berger, P. Gland, S. Singla, and F. Cucu, “Exact Analysis of TTL Cache Networks,” Performance Evaluation, vol. 79, pp. 2–23, 2014.

[10] N. Lautarit, S. Syntila, and I. Stavarakakis, “Meta Algorithms for Hierarchical Web Caches,” in Proc. of IEEE ICPCC, 2004, pp. 445–452.

[11] E. Ramadan, A. Narayanan, Z.-L. Zhang, R. Li, and G. Zhang, “Big Cache Abstraction for Cache Networks,” in Proc. Of IEEE ICDCS, 2017.

[12] A. Sasikumar, T. Zhao, I. Hou, S. Shakkottai et al., “Cache-version selection and content placement for adaptive video streaming in wireless edge networks,” arXiv preprint arXiv:1903.12164, 2019.

[13] F. Baccelli and P. Brémaud, Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences. Springer Science & Business Media, 2013, vol. 26.

[14] I. Rodriguez, A. Ferragut, and F. Paganini, “Improving Performance of Multiple-level Cache Systems,” in SIGCOMM, 2016.

[15] M. Garetto, E. Leonardi, and V. Martina, “A Unified Approach to the Convex Optimization S. Boyd and L. Vandenberghe, J. Vecer, “Dynamic Scoring: Probabilistic Model Selection Based on Utility Maximization,” in Available at SSRN, 2018.

[16] M. Massoulié, D. Towsley, D. Menasche, and Y. Tay, “A Utility Optimization Approach to Network Cache Design,” in Proc. of IEEE INFOCOM, 2016.

[17] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.

[18] J. Kyparisis, “On Uniqueness of Kuhn-Tucker Multipliers in Nonlinear Programming,” Mathematical Programming, vol. 32, no. 2, pp. 242–246, 1985.

[19] G. Tychogiorgos, A. Gkelias, and K. K. Leung, “A Non-Convex Distributed Optimization Framework and its Application to Wireless Ad-Hoc Networks,” IEEE Transactions on Wireless Communications, vol. 12, no. 9, pp. 4290–4296, 2013.

[20] A. Sundararajan, M. Feng, M. Kasbekar, and R. K. Sitaraman, “Footprint Descriptors: Theory and Practice of Cache Provisioning in a Global CDN,” in Proc. of ACM CoNEXT, 2017.

[21] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter, “Adaptsize: Orchestrating the Hot Object Memory Cache in a Content Delivery Network,” in Proc. of USENIX NSDI, 2017.

[22] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, “DNS Performance and the Effectiveness of Caching,” Networking, IEEE/ACM Transactions on, vol. 10, no. 5, pp. 589–603, 2002.