On the Asymptotic Content Routing Stretch in Network of Caches: Impact of Popularity Learning

Boram Jin, Jiin Woo, and Yung Yi

School of Electrical Engineering, KAIST, South Korea
{boramjin,jiwoo,yiyung}@kaist.ac.kr

Abstract. In this paper, we study the asymptotic average routing stretch for random content requests in a general network of caches. The key factor considered in our study is the need of learning content popularity in an on-line manner to consider time-varying changes of content popularity, where there exists a complex inter-play among (a) how long we should learn popularity, (b) how often we should change cached contents, (c) how we use learnt popularity in caching contents over the network. We model this inter-play in a broad class of caching policies, called RLP (Repeated Learning and Placement), and aim at quantifying the asymptotic routing stretch of content requests under various external conditions. Our derivation of this scaling law in the routing stretch is made under different dependence of the speed of popularity change, average routing stretch in the network of caches, the shape of the popularity distribution, and heterogeneous cache budget allocation based on nodes' geometric importance. We believe that our analytical results, even if they are asymptotic, provide additional ways and implications on understanding the delay performance of large-scale CDN (Content Distribution Network) and ICN (Information-Centric Network).

Keywords: Cache Networks · Popularity · Learning.

1 Introduction

Internet has increasingly become content-oriented and experienced the exponential traffic growth, where applications' QoS requirement becomes more and more stringent and diverse. People constantly seek for ways of adapting the Internet to such a trend. In addition to the simple effort of providing wider network pipes, it is of significant importance to build a network more content-friendly, e.g., enhancing existing CDN (Content Distribution Networks) technologies as an evolutionary approach, or proposing revolutionary architectures such as ICN (Information Centric Networking) and CCN (Content Centric Networking), see e.g., [13, 16]. In a content-oriented architecture (whether it is evolutionary or revolutionary), content caching seems to be a crucial component to reduce delay of fetching contents (from users' perspective) and/or to cut down overall traffic transport cost (from providers' perspective), often forming a group of large-scale caches (e.g. servers, access points or devices), namely a cache network.

In this paper, we aim at analytically understanding a sort of funamental limit on how long it takes for a content requester to fetch the content when caches are connected as a network. Analyzing a network of caches is known to be a daunting task in

general, since there exists a complex inter-play among the underlying (i) time-varying content popularity, (ii) content request routing, and (iii) dynamic content replacement policy. In particular, a dynamic content replacement policy is the key mechanism that plays a role of learning the popularity of contents and adaptively reconfiguring the contents in caches to the changes of active contents, where popular examples include LFU, LRU, and their variants (e.g., k-LRU and LRFU). However, analytically studying such policies even just for a *single* cache is known to be challenging [9, 10, 14], and thus a network of caches with the replacement policies is significantly challenging to analyze. To achieve our goal, despite a large degree of theoretical challenges of a network of caches, we model a network of caches under the random dynamics of content arrivals and asymptotically study the routing stretch that refers to the number of hops until a requested content is served to its requester when the size of networked caches and the number of contents scales. The metric of routing stretch in the network of caches can be used as a good approximation of the average delay in accessing the contents, provided the network load is stable so that queueing delay at a cache is regarded as an averaged constant.

Contributions. First, to provide a wide spectrum of caching strategies to the centralized planner, we first propose a highly broad class of policies, called *RLP (Repeated Learning and content Placement)*, that can contain a variety of options on popularity learning and content placement strategies. Then, we compute a lower-bound on the routing stretch by studying an ideal policy—`Oracle`—that is assumed to (i) magically obtain the knowledge on the content popularity distribution for free and (ii) (unrealistically) place contents from a requester to its original server in the decreasing order of popularity. We prove that `Oracle` is always better than any other policy $\mathcal{P}$ in the RLP class in the sense of the routing stretch, and characterize the asymptotic routing stretch performance of `Oracle` for different types of popularity distributions. Second, we develop a policy in the RLP class, called `RLP-TC` (RLP with Tilting and Cutting), that is near-optimal, i.e., its routing stretch is very close to that of `Oracle`, where by "very close" we mean that `RLP-TC` achieves the same scaling of routing stretch as `Oracle` except for a very restricted case. The smartness of `RLP-TC` lies in the idea of *tilting* and *cutting,* where tilting refers to the mechanism that modifies the learnt empirical popularity distribution into a less biased, tilted form and use this modified distribution to place contents in the caches, but we render unpopular contents uncached, i.e., cutting. We asymptotically analyze the routing stretch of `RLP-TC`.

Related work. Analyzing cache performance started from a single-cache case in the area of computer architecture and operating system [7, 14], where the main focus was on deriving asymptotic or approximate closed-form of cache hit probabilities for well-known cache policies such as LRU, LFU, and FIFO, often on the assumption of IRM (Independence Reference Model). Recently, a technique called Che's approximation [5, 9] has been applied to a simple setup, being extended to a network of caches [10, 19]. Due to analytic hardness of general topology for a network of caches, there exist work with topological restriction. Examples include the cascade [3, 18] and tree topologies [3, 20]. A few recent works started to consider general topologies [23, 24], where in [23] an algorithm called a-NET is proposed to approximate the behavior of multi-cache networks, and in [24], when a steady-state characterization of cache networks is possible.

There exist related work on asymptotic analysis of cache networks with the emphasis on throughput [2, 17] and capacity [21, 22]. In [2, 17], a dynamic content change at caches was modeled by abstracting the cache dynamics with a limited lifetime of cached content. Our work is partially inspired by [17], but we consider a more general network of caches with popularity learning.

2 Model and Problem Statement

Network. We consider a sequence of graph $\mathcal{G}(n) = (\mathcal{V}(n), \mathcal{E}(n))$, where $\mathcal{V}(n)$ is the set of nodes or caches with $|\mathcal{V}(n)| = n$, where n is the system scale size in our asymptotic study, and $\mathcal{E}(n) \subset \mathcal{V}(n) \times \mathcal{V}(n)$ describes direct connectivity between caches. Let $d_{\max}(n)$ be the maximum distance between two nodes. We let $\mathcal{C}(n)$ be the set of entire contents, and for each content $c \in \mathcal{C}(n)$, there exists its associated repository (or simply called server) that originally and permanently contains c and thus are finally accessed if a content is not fetched from an intermediate cache. Denote s_c be the content c's repository. We assume that contents are of equal size and each content $c \in \mathcal{C}$ is stored in a single server[1], say s_c, and each server $s_c \in \mathcal{S}$ is attached to a node $v_c := v_{s_c} \in \mathcal{V}$. Let $\mathcal{S}(n)$ be the set of such original content servers. Each content server is located uniformly at random in $\mathcal{V}$. Each node $v \in \mathcal{V}$ can cache a set of contents (thus node and cache can be interchangeably used throughout this paper), having the cache size $b_v(n) \geq 0$ with the network-wide cache budget $B(n)$. We assume that each cache size is equivalent across nodes, i.e., $b(n) = b_v(n) = \frac{B(n)}{n}$ for all $v \in \mathcal{V}$.

Content requests, routing and popularity. Exogenous content requests are generated at each node, which are homogeneous and independent across nodes, following a simple counting process that satisfies: the average request rate of content $c_i \in \mathcal{C} = \{c_1, c_2, \ldots\}$ is proportional to a Zipf-like distribution with parameter $\alpha > 0$:

$$p_i = K/i^{\alpha}, \tag{1}$$

where the normalizing constant K is such that $\frac{1}{K} = \sum_{i=1}^{|\mathcal{C}|} 1/i^{\alpha}$, i.e., large values of α imply higher content popularity bias. We use the notation $\mathit{rank}(c)$ to refer to the popularity ranking of content c. We abuse the notation $p_c := p_{\mathit{rank}(c)}$ to refer to the popularity distribution of content c. When a request for a content $c \in \mathcal{C}$ is generated at node v, it is forwarded along the path given by some routing algorithm [11, 12], e.g., the shortest path routing in $\mathcal{G}$, from v to the server v_c. Let d_{A} be the average distance to from each node generating a request of content c and c's repository, which we call *average to-server-distance*, where the average is taken over all pairs of request-generating node and the request's repository. In the nodes consisting of the routing path from v to v_c, the request generates HIT at w in the path if c is cached at w, and w is the first cache containing c in the path, and the content c is fetched to v via the reverse path from w to v. If no cache in the routing path has the requested content, it is MISS, in which case the content c is fetched from the original server v_c to v.

[1] This assumption does not restrict our results, because even when per-content multiple repositories exist, our asymptotic results hold as long as the number of repositories is $\Theta(1)$.

Popularity: Distribution and change. In terms of time-varying population changes, we consider the so-called *block change model*, as in [17]. In this model, during each time block, content popularity remains constant over a given $T(n)$ content requests in the entire network of size n, and then changes to some other arbitrary distribution still following Zipf-like distribution in (1), but with possibly different α, so a new time block is assume to start. Due to our intention of asymptotic approaches, our interest lies in the order of $T(n)$. In this block change model, it suffices to study our target performance metric only within the time window $[0, T(n)]$ in a single block, where the performance will be determined by a caching policy, i.e., how and how long we learn content popularity and where we place the cached contents.

Cache vs. content size. We focus on the *large content with small cache size regime* that for any given content request, the number of contents are significantly larger than the entire cache size budget $B(n)$, formally $b(n) \times d_{\max}(n) = o(|\mathcal{C}(n)|)$ and $b(n) = \Theta(1)$, where $b(n) \times d_{\max}(n)$ corresponds to an upper bound of the total possible amount of cache storage for an individual content in the entire network. Studying this regime seems quite valuable, considering the recent trend of a highly growing number of contents with the aim of reducing the content access delay at small cost of operating caches. For notational simplicity, we will drop the subscript n for all quantities that depend on n, unless confusion arises.

Performance metric: Average routing stretch. Our primary performance metric is the response delay till a content request is fetched and served. As a useful approximation of the delay, we use the *(content) routing stretch*, defined as the number of (expected) hops until it finds the desired content, i.e., HIT occurs. Formally, let random variable X_i be the routing stretch of the i-th content request (in the entire system). Then, the average routing stretch Δ of the cache network is defined as follows:

$$\Delta \triangleq \mathbb{E}\left[D\right], \quad D := \frac{1}{T(n)} \sum_{i=1}^{T(n)} X_i \tag{2}$$

which depends on the given system setups $\mathcal{G}, \alpha$, and a routing policy, as well as our controlled caching policy. Our interest is on the asymptotic characterization of Δ for large n.

3 Centralized Popularity Learning and Content Replacement Policies

3.1 Oracle Policy

This policy is the one that is assumed to obtain the true popularity statistics $[p_c : c \in \mathcal{C}]$ for free, and its algorithm is described in what follows:

`Oracle` policy

S1. Whenever any new request for a content c arrives at a node, say v, a routing path from v to c's original server v_c is ready from a given routing algorithm. Let such a routing path be a sequence of cache nodes $P_{v,c} = (v_1, v_2, \ldots, v_c)$.

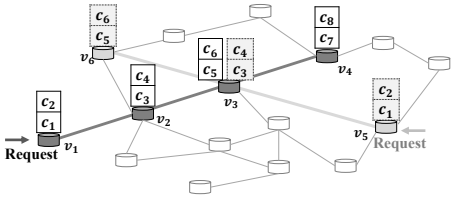


Fig. 1: Example of content placement in `Oracle`: c_i is the i-th popular content in its ranking.

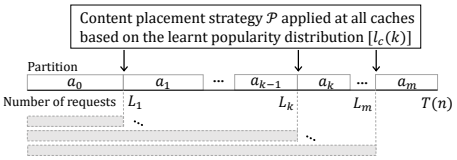


Fig. 2: Framework of online caching policy $\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$.

S2. Then, the contents are magically placed in the sequence of nodes from v_1 to v_c, with the decreasing order of content popularity (which is given for free), where each node w in the routing path can cache the $b = B/n$ number of contents for the corresponding request.

To illustrate, we consider the example in Fig. 1, where suppose that $b = 2$, i.e., each node can cache 2 contents. Assume that we have a new content request generated at v_1, the content's original server is at v_4, and its priori given routing path is (v_1, v_2, v_3, v_4). Then, we cache the contents from v_1 to v_4 in the decreasing order of popularity, two contents at each node in the routing path, thus total 8 contents in the path. Note that `Oracle` is *unrealistic* due to the following reasons: In addition to magically-given knowledge on the true popularity statistics, there may be the case when a cache should store the contents beyond its given cache size. For example, as illustrated in Fig. 1, for a request generated at v_5 whose routing path to the original sever is (v_5, v_3, v_6), v_3 should store the contents c_3, c_4, whereas v_3 is supposed to store the contents c_5, c_6 for the request by v_1. We allow this violation of cache size limit in `Oracle`, because we plan to use `Oracle` as a policy providing a lower bound of the routing stretch.

3.2 RLP Class and RLP-TC (Tilting and Cutting)

We now consider a class of polices, called RLP (Repeated Learning and Placement), where a policy in the RLP class has repeated steps for popularity learning and content placement. We claim that the RLP class is highly general so as to include any possible policies that mix popularity learning and configuring cache contents in a centralized manner.

We now elaborate on the class of `RLP` policies. An `RLP` policy, $\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$, is parameterized by (i) the number of repetition steps m, (ii) m-dimensional vector $\boldsymbol{a}(m) = [a_i : i = 0, 1, \ldots, m]$, where $[a_i : i = 0, 1, \ldots, m]$ defines the $m+1$ sequential temporal partitions from the first to $T(n)$ requests, and (iii) content placement strategy $\mathcal{P}$. See Fig. 2 for a pictorial description of an $\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$. Note that $\sum_{i=0}^{m} a_i = T(n)$. Let $L_k = \sum_{i=0}^{k-1} a_i$, i.e., the aggregate number of requests until the partition a_{k-1}. Then, the partition a_k turns out to be the number of requests from when the system receives (L_k+1)-th request to L_{k+1}-th request. The basic idea of the $\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$ is that at each partition a_k we first learn and estimate the content popularity distribution using the L_k requests, and use the learnt popularity in the content placement strategy $\mathcal{P}$. Examples

of content placement strategy $\mathcal{P}$ include a random strategy of simply placing contents uniformly at random and a popularity-proportional strategy where the probability that a content is placed in a cache is proportional to the (learnt) popularity.

$\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$ is formally described by the following recursive procedures, that specify what have to be done at the start of each partition a_k, $k = 0, 1, \ldots$:

$\texttt{RLP}(\boldsymbol{a}, m, \mathcal{P})$

At partition a_0: Contents are placed uniformly at random at each cache of size $b = B/n$.

At partition a_k:

- *Popularity learning phase:* The system learns the popularity distribution $[l_c(k) : c \in \mathcal{C}]$ by computing the following empirical distribution:

$$l_c(k) = \frac{\sum_{j=1}^{L_k} Y_c^j}{L_k},$$

  where $Y_c^j = 1$ if j^{th} request is for content c, and 0 otherwise.
- *Content placement phase:* Then, the content placement strategy $\mathcal{P}$ is applied at all caches based on the learnt popularity distribution $[l_c(k) : c \in \mathcal{C}]$, and new requests over the partition a_k are served.

Note that at partition a_0 it is natural to employ a uniformly random policy because there is no knowledge about popularity obtained in the past. One of trivial policies belonging to the RLP class is $\texttt{RLP}(\boldsymbol{a}, 0, \texttt{RANDOM})$ corresponding to the policy that caches the contents uniformly at random in the network without any learning of content popularity. Our goal is to achieve short routing stretch by intelligently choosing m, $\boldsymbol{a}(m)$, and $\mathcal{P}$.

We now propose a policy in the RLP class, called RLP-TC (RLP with *tilted popularity distribution with cutting*. Since RLP-TC is a policy in the RLP class, its unique feature is characterized by (i) content placement strategy $\mathcal{P}$ and (ii) a construction of $\boldsymbol{a}(m)$ (see Section 3.2). As a content placement strategy $\mathcal{P}$, we re-manufacture the learnt empirical distribution into a "tilted distribution" and use it for content placement, which we call TC (Tilted learnt popularity with Cutting).

Regimes: Speed of popularity change. Prior to describing RLP-TC, we first describe three different regimes with respect to how fast content popularity changes: *Fast*, *Normal*, and *Slow*, which, in practice, may differ depending on the content categories [25]. This classification is used to describe the caching policies, present our analytical results and their interpretations. We define three regimes of $T(n)$ as follows:

$$\begin{cases} \textsf{FAST} & \text{if } \Theta(\log d_{\mathrm{A}}) \le T(n) \le \Theta(d_{\mathrm{A}} \log d_{\mathrm{A}}), \\ \textsf{NORMAL} & \text{if } \Theta(d_{\mathrm{A}} \log d_{\mathrm{A}}) < T(n) < \Theta(d_{\mathrm{A}}^2 \log d_{\mathrm{A}} \cdot M^2), \\ \textsf{SLOW} & \text{if } \Theta(d_{\mathrm{A}}^2 \log d_{\mathrm{A}} \cdot M^2) \le T(n), \end{cases} \tag{3}$$

where recall d_{A} is the average to-server-distance, and $M = M(\alpha)$ is such that:

$$\frac{1}{M} = \sum_{i=1}^{d_{\mathrm{A}}} \frac{1}{i^{\alpha/2}} = \begin{cases} \Theta(1) & \text{if} \quad \alpha > 2, \\ \Theta(\log d_{\mathrm{A}}) & \text{if} \quad \alpha = 2, \\ \Theta(d_{\mathrm{A}}^{1-\alpha/2}) & \text{if} \quad 2 > \alpha > 1. \end{cases} \tag{4}$$

Note that our classification differs in NORMAL and SLOW regimes depending on M, which also relies on the popularity bias parameter α.

Policy description. We first describe and explain RLP-TC policy, followed by its rationale.

RLP-TC $=$ RLP$(\boldsymbol{a}, m,$ TC$)$ policy

INPUT: $T(n)$ and α.

$\circ$ ***Construction of*** m ***and*** $\boldsymbol{a}$***:*** We first choose a_0 with its dependence on $T(n)$ as follows:

$$a_0 = \begin{cases} T(n) & \text{if} \quad \textsf{FAST}, \\ \Theta(\log d_{\mathrm{A}}) & \text{if} \quad \textsf{NORMAL and SLOW}, \end{cases} \tag{5}$$

Then, for $\Theta(d_{\mathrm{A}} \log d_{\mathrm{A}}) < T(n)$, choose $a_1 = \max\{o(a_0), \Theta(1)\}$, and select m:

$$m = \log_r \left(1 - \frac{(1-r)(T(n) - a_0)}{a_1}\right), \tag{6}$$

where

$$r = 1 - \frac{a_1}{T(n)} \quad (< 1). \tag{7}$$

Then, the remaining sequence $(a_2, a_3, \ldots, a_m)$ is constructed by the geometric series, starting from a_1, with the common ratio r, i.e., $a_k = a_1 r^{k-1}$, $k = 1, \ldots, m$.

$\circ$ TC ***strategy:*** In the general RLP policy description, at each step a_k, we apply the following procedures to the *content placement phase*:

S1. *Construction of tilted popularity distribution.* Using the empirical distribution $[l_c(k) : c \in \mathcal{C}]$, we compute the following tilted distribution $[\hat{l}_c(k) : c \in \mathcal{C}]$:

$$\hat{l}_c = \begin{cases} \frac{\sqrt{l_c}}{\sum_{c=1}^{\hat{i}} \sqrt{l_c}} & \text{if} \quad \textit{rank(c)} \le \hat{i}, \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

where $\hat{i} = b \cdot d_{\mathrm{A}}$ and recall that *rank(c)* is the popularity ranking of content c.

S2. At each cache, b $(= B/n)$ contents are randomly selected according to the distribution $[\hat{l}_c(k) : c \in \mathcal{C}]$ without duplication of contents.

Policy explanation. We present the rationale of RLP-TC. First, the constructed $\boldsymbol{a}$ and m depend on a given $T(n)$ and the popularity bias α, where a_0's dependence is of significant importance. As observed in (5), the length of a_0 decreases in NORMAL and SLOW rather than FAST, because for larger $T(n)$, more chances to learn the popularity are allowed, whereas for smaller $T(n)$, the initial learning becomes more crucial. In NORMAL and SLOW, the remaining steps $(a_1, a_2, \ldots, a_m)$ as well as the total number of iterations m are chosen as a geometric series such that their sum equals to $T(n)$, as seen in (6) and (7). Second, as a content placement strategy, the proposed TC strategy first re-manufactures the learnt popularity distribution into a *tilted form with cutting,* as in (8) at each step k. It means that we nullify the distribution of unpopular contents, where we maintain the popularity only up to the ranking index $\hat{i} = b \cdot d_{\mathrm{A}}$, and re-normalize the distribution (more specifically taking the square root, i.e., $\sqrt{l_c}$, as seen in (8)). Then, we randomly place the contents according to the computed tilted distribution (**S2**). To intuitively understand, take an example of three contents, where the originally learnt distribution in step k is $(l_c(k)) = (0.7, 0.2, 0.1)$ with $\hat{i} = 2$. Then, in our tilted distribution with cutting, we have $(\hat{l}_c(k)) = (0.65, 0.35, 0)$. We will explain how this helps in achieving short routing stretch next.

Rationale of tilting with cutting. A natural way is to directly use the empirically learnt popularity distribution with which contents are randomly placed at each cache. However, what we do is to use a tilted distribution $[\hat{l}_c(k)]$, such that $[\hat{l}_c(k) \propto \sqrt{l_c(k)} : c \in \mathcal{C}]$. The key effects of tilting with cutting are summarized in what follows: (i) *tilting:* making the popularity distribution less biased and (ii) *cutting:* making unpopular contents uncached. If we consider only a single, stand-alone cache, just the empirical distribution-based placement might be enough. However, in the network of caches, the performance of our interest, which is a routing stretch, is a non-trivial complex function of coupled behaviors among the caches in a given routing path. This non-trivial relationship requires us to re-consider the obtained empirical popularity distribution and also effectively use the available rooms for caching whose size is strictly smaller than the number of contents. This motivation leads us to tilt the empirical distribution with unpopular contents excluded from caches.

Why square root in tilting? The remaining question in tilting, is why the choice of $\sqrt{l_c(k)}$ is made for the obtained empirical distribution $l_c(k)$? Just for simplicity of exposition, assume $b = 1$, i.e., each node can cache only one content, and also the to-server-distance d_{A} is large. We now consider a cache placement strategy under which content c_i (whose popularity distribution is p_i) is cached in each cache with probability q_i. For large $T(n)$, the expected routing stretch Δ for d_{A} roughly becomes:

$$\Delta = \sum_{i=1}^{|\mathcal{C}|} p_i \cdot \frac{1}{q_i} = \Big(\sum_{i=1}^{|\mathcal{C}|} p_i \frac{1}{q_i}\Big)\Big(\sum_{i=1}^{|\mathcal{C}|} q_i\Big) \geq \Big(\sum_{i=1}^{|\mathcal{C}|} {p_i}^{\frac{1}{2}}\Big)^2,$$

where $1/q_i$ is the average stretch from a requester to the cached node of content c_i, and the last inequality comes from the Cauchy-Schwarz inequality. In Cauchy-Schwarz inequality, it is widely known that the equality holds if and only if there is some constant k such that $p_i \frac{1}{q_i} = k \cdot q_i$ for all i. Therefore, Δ is minimized when $q_i \propto i^{-\frac{\alpha}{2}}$, and the minimum value is $\big(\sum_{i=1}^{|\mathcal{C}|} {p_i}^{\frac{1}{2}}\big)^2$. This is why $\sqrt{l_i}$ is selected for TC where l_i goes to

p_i for sufficiently large $T(n)$. Note that a special case when $q_i = p_i$ corresponds to the case utilizing content popularity distribution directly for the cache placement, and $\Delta = |\mathcal{C}|$.

4 Analysis: Routing Stretch

We now present our main results on the average routing stretch Δ, as addressed in (2), for `Oracle` and `RLP-TC`, where all the proofs are presented in our technical report [15]. To make our analysis of Δ tractable, we first rewrite Δ as the average over a random to-server-distance d_{tsd} from a content requester to the corresponding content server (where the randomness comes from the location of content requesters and the corresponding content servers) as follows:

$$\Delta = \mathbb{E}[\Delta(d_{\text{tsd}})] = \sum_{d} f_{\text{tsd}}(d)\Delta(d) \tag{9}$$

where, to abuse the notation, $\Delta(d)$ be the "expected" routing stretch when to-server-distance is d, where the expectation is taken with respect to the randomness in the contents and content caching policy, and $f_{\text{tsd}}(d)$ is the distribution of d.[2] Note that having a closed form $f_{\text{tsd}}(d)$ is challenging and thus makes the routing-stretch analysis hard, because $f_{\text{tsd}}(d)$ depends on the given topology $\mathcal{G}$ and the underlying routing algorithm. For example, even for a Erdös−Rényi (ER) random graph, which is one of the simplest random graphs, when the shortest path routing algorithm is used, $f_{\text{tsd}}(d)$ is still unknown. Thus, to purely focus on our interest, we use Jensen's inequality and obtain:

$$\Delta = \mathbb{E}\Big[\Delta(d_{\text{tsd}})\Big] \leq \Delta\Big(\mathbb{E}[d_{\text{tsd}}]\Big) = \Delta(d_{\text{A}}), \tag{10}$$

where recall that d_{A} is the average to-server-distance, and we now consider $\Delta(d_{\text{A}})$ as our major metric to analyze. To differentiate from $\Delta(d)$ for any given d, we often call $\Delta(d_{\text{A}})$ *average stretch upper-bound*, or simply *stretch upper-bound*.

4.1 `Oracle`

This is formally presented in Theorems 1 which states that it is optimal in the sense that `Oracle` has shorter routing stretch than any other policy in the RLP class, and in Theorem 2 which presents the asymptotic average routing stretch of `Oracle`.

Theorem 1. *Let $D^{\mathcal{O}}$ and $D^{\mathcal{A}}$ be the random routing stretches (as defined in* (2)*) of* `Oracle` *and an arbitrary policy $\mathcal{A}$ in the RLP class, respectively. Then, the following stochastic dominance of* `Oracle` *holds: $D^{\mathcal{O}} \leq_{st} D^{\mathcal{A}}$, which means $\mathbb{P}\Big[D^{\mathcal{O}} > x\Big] \leq \mathbb{P}\Big[D^{\mathcal{A}} > x\Big]$, for any $x \geq 0$.*

Theorem 2. *For a given to-server-distance d between a pair of a content requester and its server, the average routing stretch $\Delta(d)$ and the stretch upper-bound $\Delta(d_A)$ of* `Oracle` *scale as those in Table 1.*

[2] d is also a random variable since a chosen content is also random.

Table 1: Routing stretch: `Oracle`

Popularity	$\Delta(d)$	$\Delta(d_{\mathrm{A}})$
$2 < \alpha$	$\Theta(1)$	$\Theta(1)$
$\alpha = 2$	$\Theta(\log d)$	$\Theta(\log d_{\mathrm{A}})$
$1 < \alpha < 2$	$\Theta(d^{2-\alpha})$	$\Theta(d_{\mathrm{A}}^{2-\alpha})$
$0 < \alpha \leq 1$	$\Theta(d)$	$\Theta(d_{\mathrm{A}})$

Table 2: Stretch UBs of `RLP-TC-OL` and `RLP-TC`

Regime		`RLP-TC-OL`	`RLP-TC`
FAST		$\Theta(d_{\mathrm{A}})$	$\Theta(d_{\mathrm{A}})$
NORMAL		$\Theta\left(\sqrt{d_{\mathrm{A}}^3 \frac{\log d_{\mathrm{A}}}{T(n)}}\right)$	$\Theta\left(d_{\mathrm{A}}^2 \frac{\log d_{\mathrm{A}}}{T(n)}\right)$
SLOW	$T(n) < \Theta(d_{\mathrm{A}}^3 \log d_{\mathrm{A}})$	$\Theta\left(\sqrt{d_{\mathrm{A}}^3 \frac{\log d_{\mathrm{A}}}{T(n)}}\right)$	$\Theta(\frac{1}{M^2})$
	$T(n) \geq \Theta(d_{\mathrm{A}}^3 \log d_{\mathrm{A}})$	$\Theta(\frac{1}{M^2})$	$\Theta(\frac{1}{M^2})$

Thanks to Theorem 1, the result of asymptotic routing stretches in Table 1 provides lower bounds of $\Delta(d)$ of any policy in the RLP class. As expected, as α decreases, we lose the power of caching, thus leading to the increase of routing stretch. When $\alpha > 2$, the stretch is order-wise optimal (i.e., a constant order), and only up to $\alpha = 2$, the stretch is sub-polynomial. Note that when $0 < \alpha \leq 1$, the caching gain vanishes, so requiring to reach the corresponding original server. We now seek to find a policy in the RLP class whose performance is close to that of `Oracle`, if any.

4.2 `RLP-TC`

We now present the result on the stretch bound $\Delta(d_{\mathrm{A}})$ of `RLP-TC` in Theorem 3. We will focus only on the case when $\alpha > 1$, because even the lower-bound provided by `Oracle` proves that there is no caching gain for $\alpha \leq 1$, see Theorem 1.

Theorem 3. *For $\alpha > 1$, the upper-bound of routing stretch $\Delta(d_A)$ for* `RLP-TC` *scales as follows: With high probability,*

$$\Delta(d_A) = \begin{cases} \Theta(d_A) & \textit{if} \quad \textsf{FAST}, \\ \Theta\left(d_A^2 \frac{\log d_A}{T(n)}\right) & \textit{if} \quad \textsf{NORMAL}, \\ \Theta(\frac{1}{M^2}) & \textit{if} \quad \textsf{SLOW}. \end{cases} \tag{11}$$

Note that the performance of `RLP-TC` depends on $T(n)$, as expected. In FAST, $\Delta(d_{\mathrm{A}}) = \Theta(d_{\mathrm{A}})$, i.e., there is no gain due to the lack of time to learn and apply such a learning result to the content placement. In NORMAL, $\Delta(d_{\mathrm{A}})$ decreases as $T(n)$ increases, because the repeated learning process helps where we have enough time to learn the popularity and use such knowledge in placing contents, until $\Delta(d_{\mathrm{A}})$ reaches $\Theta(1/M^2)$ in (11) at the threshold $T(n) = \Theta(d_{\mathrm{A}}^2 \log d_{\mathrm{A}} \cdot M^2)$ in (3). After this threshold, i.e., SLOW, the repeated learning and placement do not help in reducing routing stretch, so as to keep the stretch $\Theta(1/M^2)$. Since M is inversely proportional to α as in (4), we can conclude that a system efficiently utilizes the chance of learning with relatively smaller $T(n)$ for the cases of having highly biased content popularity. In SLOW, `RLP-TC` is ***near-optimal,*** since it achieves $\Delta(d_{\mathrm{A}})$ as `Oracle` in Table 1 for $\alpha > 1$ except the case $\alpha = 2$, we have an order-wise difference $\Theta(\log d_{\mathrm{A}}) = O(\log n) = o(n)$ between $\Theta(\log^2 d_{\mathrm{A}})$ in `RLP-TC` and $\Theta(\log d_{\mathrm{A}})$ in `Oracle`.

Application to Power-law and Erdös−Rényi graphs. As case studies, we now apply Theorem 3 to two popular random graphs: Power-law (PL) and ER graphs, which have

well-known results for d_{A}, so that we are able to obtain more concrete stretch representation. In the PL graph, the fraction of nodes with degree i is proportional to $1/i^{\gamma}$ for some constant $\gamma > 0$. If the average degree is strictly greater than 1, and $2 < \gamma < 3$, it is known that the average to-sever-distance d_{A} under the shortest path routing is $d_{\mathrm{A}} = \Theta(\log n/ \log\log n)$ [6]. The ER-graph is constructed by randomly connecting two nodes with some probability, say p. If np is of the order $\log n$, then the graph almost surely contains a giant component of size of order n connected with high probability, and it is known by [8] that the average to-server-distance under the shortest path routing is $d_{\mathrm{A}} = \Theta(\log n/ \log np)$. Using those facts about d_{A} under two example graphs and applying d_{A} to Theorem 3, for given $T(n)$ and α, we can obtain the stretch upper bound. For example, suppose we consider the case $\alpha > 2$ (e.g., Youtube [4]) and $d_{\mathrm{A}} = \Theta(\log n/ \log\log n)$, where the ER graph with $np = \Theta(\log n)$ and PL graph with $2 < \gamma < 3$ and the average degree strictly greater than 1. Under three regimes, we get:

$$\Delta(d_{\mathrm{A}}) = \begin{cases} \Theta\left(\frac{\log n}{\log\log n}\right) & \text{if} \quad \mathsf{FAST}, \\ \Theta\left(\frac{\log^2 n}{\log\log n} \cdot \frac{1}{T(n)}\right) & \text{if} \quad \mathsf{NORMAL}, \\ \Theta(1) & \text{if} \quad \mathsf{SLOW}. \end{cases}$$

Thus, for example, we see that under FAST, the routing stretch has order $\Theta(\frac{\log n}{\log\log n})$, and inversely proportional to $T(n)$ during NORMAL. Under SLOW, the routing stretch is order-optimal in both cases in PL and ER graphs , i.e., the same as `Oracle`.

Trade-off between learning complexity and efficiency. We now purely study the impact of repeated learning, by considering a policy `RLP-TC-OL` (`RLP-TC` with One-time Learning). `RLP-TC-OL` is the same as `RLP-TC` except that we have $\boldsymbol{a} = (a_0, a_1)$, where a_0 and a_i are as in `RLP-TC`, i.e., we learn the popularity during a_0 only once, and use the learnt popularity during the remaining $T(n)$ without no further learning. This comparison, as presented in Table 2, gives us interesting messages on the impact of repeated learning and the trade-off between system overheads and the routing stretch. Note that as m increases, contents at a cache should be more often re-organized, incurring an increasing number of file copies and usage of network bandwidths. Both policies' stretch start at the order $\Theta(d_{\mathrm{A}})$ because of the lack of learning time in both policies. However, in NORMAL, `RLP-TC` decreases more rapidly because of $T(n)$ in the denominator of $\Delta(d_{\mathrm{A}})$ as shown in Table 2 (rather that $\sqrt{T(n)}$), and thus the effect of repeated learning becomes visible. In SLOW, both schemes again perform similarly, because, in presence of enough time to learn and play, only a long period of initial learning time, i.e., a_0 is enough to achieve short routing stretch without need of repeated learning.

5 Simulation Results

5.1 Setup

In this section, we present simulation results to study the practical relevance of `RLP-TC` to LFU and LRU, where we also plot the result of `Oracle` to show the fundamental limit. To slightly elaborate on our implementation of tested algorithms, first in `Oracle`,

we exactly follow the description in Section 3.1 with allowing the cache size of some intermediate nodes to be larger than the given limit. In the RLP class, we employ two simplified versions, marked as: (i) RLP-TC: RLP-TC with $a_0 = 1$, and $a_i = 1$ for $1 \leq i \leq T(n)$ (update the popularity statistics for each request, and globally change the cache contents in the network), and (ii) RLP-TC(5,50): RLP-TC with $a_0 = 5$ and $a_1 = 45$ (i.e., just two-time learning during a given $T(n)$). The result of RLP-TC(5,50) helps in understanding the impact of limited number of popularity learning chances. The popularity follows Zipf-like distribution in (1) with various values of popularity bias α.

We consider three topologies in our simulations as described in what follows:

- *Line topology:* This is a simple line topology consisting of 10 nodes, where $|\mathcal{C}| = 1000$ and $b = 5$. All content requests arrive at the first node based on the content popularities and the last cache is connected to servers of all contents, and unresolved requests are forwarded to the next caches under the shortest path routing.
- *Tree topology;* We consider a binary tree topology with total 1023 nodes, where the height is 10, a root node has all contents as a server, and all requests arrive at the bottom leaves. Thus the maximum routing stretch from the content-requested node to the server is 10).
- *AS (Autonomous System) topology:* We consider tree AS topologies: Cogent (USA-Europe), Colt Telecom (Europe), and TW Telecom (USA) from [1], as seen in Fig. 3. We conducted an off-line processing to extract the topological features of these three topologies, presented in Table 3, which are highly heterogeneous except for the average degree.

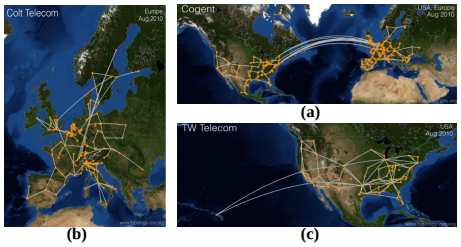


Fig. 3: AS topologies of (a) Cogent (Europe-USA), (b) Colt Telecom (Europe), and (c) TW Telecom (USA) [1]

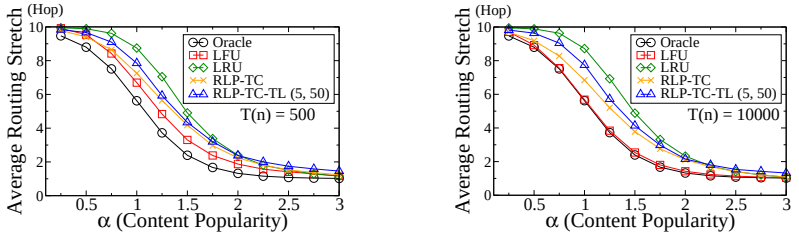
Table 3: Simulation environments

Topology	Cogent	Colt Tel.	TW Tel.
N	197	153	76
Avg. deg., d_{A}	2.49,10.4	2.50,8.24	3.08,3.21
$\|\mathcal{C}\|$, b	3000, 5		
$\hat{i} = \min[b \cdot d_{\mathrm{A}}, \|\mathcal{C}\|]$	50	40	15

We construct the simulation environment such that $b \cdot d_{\mathrm{A}} \ll |\mathcal{C}|$, based on the recent trend of explosive increase in the number of contents. To get simulation results, we perform 40 times of random instances.

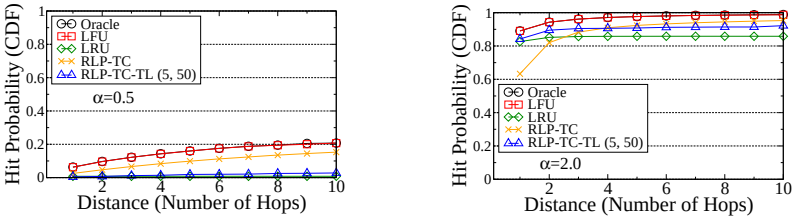
5.2 Results

Line topology. We compare the routing stretches of RLP-TC, and RLP-TC-TL (5, 50) with LRU and LFU as shown in Fig. 4 for $T(n) = 500$, and 10000. Oracle has the lowest stretch. RLP-TC, and RLP-TC-TL (5, 50) performs between LFU and LRU, where



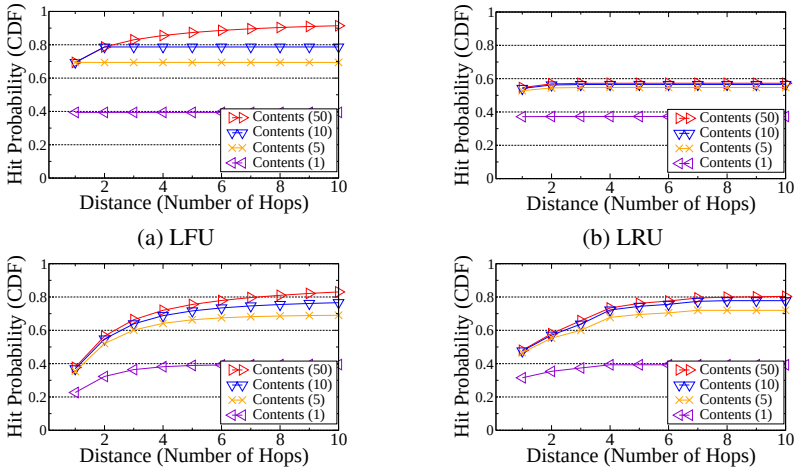
(a) $T(n) = 500$ (b) $T(n) = 10000$

Fig. 4: Line topology. Average routing stretches of five policies (`Oracle`, LFU, LRU, `RLP-TC` and `RLP-TC-TL` (5, 50)) for various α at $T(n)$ =500 and 10000.



(a) $\alpha = 0.5$ (b) $\alpha = 2.0$

Fig. 5: Line topology. Hit probability (CDF) of five policies (`Oracle`, LFU, LRU, `RLP-TC`, and `RLP-TC-TL` (5, 50)) for high ranked 50 contents over α= 0.5, and 2.0.

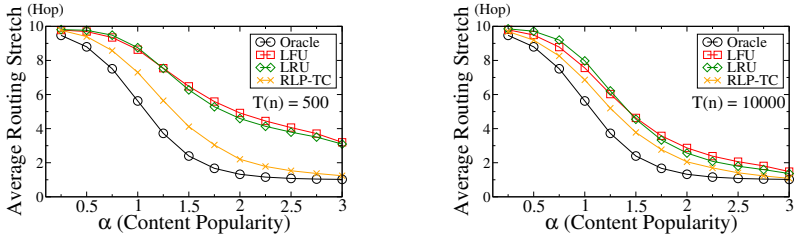


(a) LFU (b) LRU

(c) `RLP-TC` (d) `RLP-TC-TL` (5, 50)

Fig. 6: Line topology. Hit probability (CDF) of four policies (LFU, LRU, `RLP-TC` and `RLP-TC-TL` (5, 50)) for high ranked 1, 5, 10, and 50 contents for $\alpha = 1.5$.

we see that (i) a small number of popularity learning at the starting period of the system is highly beneficial (at least under our setting), and (ii) our analysis of centralized `RLP-TC` algorithms can be a good approximation of the one between LFU and LRU. Note that it is natural that LFU outperforms LRU, since LRU has more strength when the content configurations are dynamically changing. Figs. 5 and 6 show on average



(a) $T(n) = 500$ (b) $T(n) = 10000$

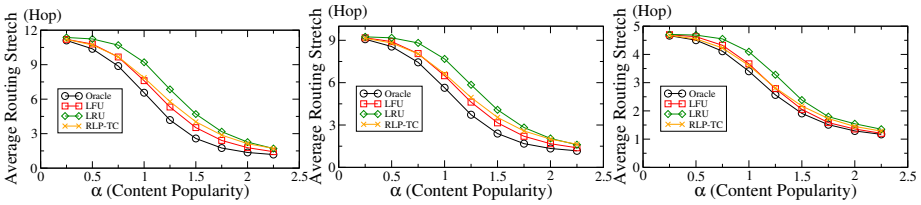
Fig. 7: Tree topology. Average routing stretches of four policies (`Oracle`, LFU, LRU, and `RLP-TC`) for various α at $T(n)$ =500 and 10000.

when the requested contents experience HIT. `Oracle` and LFU show the similar cumulative distribution function (CDF) of hit probability for 50 top-ranked contents, since LFU operates so as to autonomously place the contents as an ascending order of content popularity based on the history of requested contents at each node. Although the hit probability of `RLP-TC` for the first node is lower than that of LRU because `RLP-TC` probabilistically selects the contents based on the estimated popularity, at 10-th node, the sum hit probability (CDF) of `RLP-TC` exceeds that of LRU, where `RLP-TC` smartly utilizes the tilted distribution with cutting. Thus, in LRU, there is no increment of the sum hit probabilities (CDF) between 10 and 50 top-ranked contents due to caching unpopular contents (over the rank 50). Despite less active learning in `RLP-TC-TL` (5, 50), it is able to detect the relatively highly popular contents, achieving the high hit probability at the first node, but due to the lack of information, `RLP-TC-TL` (5, 50) does not cache contents within ranked 10-th to 50-th contents in the backside caches in a line, so as to have the higher routing stretch than `RLP-TC`.

Tree and AS topologies. First in tree topology, we compare the performance of `RLP-TC` ($a_i = 1$ for $0 \leq i \leq T(n)$) with LFU and LRU when content popularity follows Zipf-like distribution with $|\mathcal{C}| = 1000$ and $b = 5$. As shown in Fig 7, average stretches of LFU and LRU are similar to that of `RLP-TC` when $T(n) = 10000$, and `RLP-TC` outperforms LFU and LRU for $T(n) = 500$ because `RLP-TC` utilizes the gathered information and replaces cached contents based on information for each request. In AS topologies, we perform the simulation during 100000 slots where we focus on SLOW regime assuming that popularity distribution is a priori given to `RLP-TC`. For each test, we first place content servers uniformly at random, and a content request arrives at a cache with probability 0.5 at the beginning of the time slot, and unresolved requests are forwarded to the next cache under the shortest path routing. Fig. 8 shows the results of various caching policies over three topologies. We compare the performance of `RLP-TC` with dynamic caching strategies LFU and LRU. Figure 8 shows the absolute (average) routing stretch performances of two dynamic cache replacement policies, LRU and LFU, compared to the `RLP-TC`, for three graph topologies. As done in the analysis earlier, the average routing stretch in y-axis corresponds to the number of hops. Table 4 shows the normalized average routing stretches of LFU and LRU by that of `RLP-TC`. We observe that `RLP-TC`'s routing stretch has good match in those of LFU and LRU (on average, about 5.2% and 9.6% differences for LFU and LRU, respectively). Note that LFU is known to perform better than LRU with higher implementation complexity. From our

Table 4: AS topology. Average routing stretches of LFU and LRU normalized by that of RLP-TC

Topology α	0.5	1	1.5	2
Cogent (LFU)	1.017	0.967	0.874	0.836
Cogent (LRU)	1.048	1.170	1.167	1.063
Colt (LFU)	1.011	0.979	0.891	0.839
Colt (LRU)	1.041	1.158	1.152	1.026
TW (LFU)	1.017	1.016	0.945	0.937
TW (LRU)	1.030	1.135	1.097	1.066



(a) Cogent (b) Colt (c) TW

Fig. 8: AS topology. Average routing stretch performance.

simulation results, our analysis considering static cache policies can be used to predict the large-scale cache networks' average routing stretch performance.

6 Conclusion

We presented asymptotic analysis of the routing stretch of large-scale cache networks. We focused on quantitatively understanding the relation between content popularity and average to-server-distance, as well as the impact of learning and cache sizing heterogeneity. We studied the asymptotic routing stretch of cache networks under on-line repeated learning and content placement policy. We also derived that this scaling routing stretch is made under different dependence of the speed of popularity change, average to-server-distance in the network of caches, and the shape of the popularity distribution.

Ackknowledgment. This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00170, Virtual Presence in Moving Objects through 5G and No.2016-0-00160,Versatile Network System Architecture for Multi-dimensional Diversity).

References

1. The Internet topology zoo. http://www.topology-zoo.org/dataset.html
2. Azimdoost, B., Westphal, C., Sadjadpour, H.R.: On the throughput capacity of information-centric networks. In: Proc. ICT (2013)
3. Carofiglio, G., Gallo, M., Muscariello, L., Perino, D.: Modeling data transfer in content-centric networking. In: Proc. ITC (2011)

4. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.Y., Moon, S.: Analyzing the video popularity characteristics of large-scale user generated content systems. IEEE Transactions on Networking **17**(5), 1357–1370 (2009)
5. Che, H., Wang, Z., Tung, Y.: Analysis and design of hierarchical Web caching systems. In: Proc. IEEE Infocom (2001)
6. Chung, F., Lu, L.: The average distances in random graphs with given expected degrees. Proc. National Academy of Sciences **99**(25), 15879–15882 (2002)
7. Dan, A., Towsley, D.: An approximate analysis of the LRU and FIFO buffer replacement schemes. Performance Evaluation Review **18**(1), 143–152 (1990)
8. Draief, M., Massouli, L.: Epidemics and rumours in complex networks. Cambridge University Press (2010)
9. Fricker, C., Robert, P., Roberts, J.: A versatile and accurate approximation for LRU cache performance. In: Proc. ITC (2012)
10. Garetto, M., Leonardi, E., Martina, V.: A unified approach to the performance analysis of caching systems. ACM TOMPECS **1**(3), 12 (2016)
11. Gitzenis, S., Paschos, G.S., Tassiulas, L.: Asymptotic laws for joint content replication and delivery in wireless networks. IEEE Transactions on Information Theory **59**(5), 2760–2776 (2013)
12. Ioannidis, S., Yeh, E.: Adaptive caching networks with optimality guarantees. In: Proc. ACM SIGMETRICS (2016)
13. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: Proc. ACM CoNext (2009)
14. Jelenković, P.: Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities. The Annals of Applied Probability **9**(2), 430–464 (1999)
15. Jin, B., Woo, J., Yi, Y.: On the asymptotic content routing stretch in network of caches: Impact of popularity learning. Tech. rep., KAIST, South Korea (2018), `http://lanada.kaist.ac.kr/pub/cache.pdf`
16. Koponen, T., Chawla, M., Chun, B.G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I.: A data-oriented (and beyond) network architecture. In: Proc. ACM SIGCOMM (2007)
17. Moharir, S., Ghaderi, J., Sanghavi, S., Shakkottai, S.: Serving content with unknown demand: the high-dimensional regime. In: Proc. ACM SIGMETRICS (2014)
18. Muscariello, L., Carofiglio, G., Gallo, M.: Bandwidth and storage sharing performance in information centric networking. In: Proc. ACM SIGCOMM workshop on Information-centric networking (2011)
19. Neglia, G., Carra, D., Michiardi, P.: Cache policies for linear utility maximization. In: Proc. IEEE Infocom (2017)
20. Psaras, I., Clegg, R.G., Landa, R., Chai, W.K., Pavlou, G.: Modelling and evaluation of CCN-caching trees. In: Proc. NETWORKING. Springer (2011)
21. Qiu, L., Cao, G.: Cache increases the capacity of wireless networks. In: Proc. IEEE Infocom (2016)
22. Qiu, L., Cao, G.: Popularity aware caching increases the capacity of wireless networks. In: Proc. IEEE Infocom (2017)
23. Rosensweig, E., Kurose, J., Towsley, D.: Approximate models for general cache networks. In: Proc. IEEE Infocom (2010)
24. Rosensweig, E.J., Menasche, D.S., Kurose, J.: On the steady-state of cache networks. In: Proc. IEEE Infocom (2013)
25. Sikdar, S., Chaudhary, A., Kumar, S., Ganguly, N., Chakraborty, A., Kumar, G., Patil, A., Mukherjee, A.: Identifying and characterizing sleeping beauties on youtube. In: Proc. ACM CSCW (2016)