Performance of Probabilistic Caching and Cache Replacement Policies for Content-Centric Networks

Saran Tarnoi[†‡], Kalika Suksomboon[§], Wuttipong Kumwilaisak[¶], and Yusheng Ji[†‡]
[†] Department of Informatics, The Graduate University for Advanced Studies, Tokyo, Japan
[‡] National Institute of Informatics, Tokyo, Japan
[§] KDDI R&D Laboratories, Inc., Saitama 356-8502, Japan
[¶] King Mongkut's University of Technology Thonburi, Bangkok, Thailand
Email: {saran, kei}@nii.ac.jp, ka-suksomboon@kddilabs.jp, wuttipong.kum@kmutt.ac.th

***Abstract*—The Content-Centric Networking (CCN) architecture exploits a universal caching strategy whose inefficiency has been confirmed by research communities. Various caching schemes have been proposed to overcome some drawbacks of the universal caching strategy but they come with additional complexity and overheads. Besides those sophisticated caching schemes, there is a probabilistic caching scheme that is more efficient than the universal caching strategy and adds a modest complexity to a network. The probabilistic caching scheme was treated as a benchmark and the insights into its behavior have never been studied despite its promising performance and feasibility in practical use. In this paper we study the probabilistic caching scheme by means of computer simulation to explore the behavior of the probabilistic caching scheme when it works with various cache replacement policies. The simulation results show the different behavioral characteristics of the probabilistic caching scheme as a function of the cache replacement policy.**

Keywords*—Content-centric networking, information-centric networking, caching strategy, cache replacement policy.

I. Introduction

Content-Centric Networking (CCN) [1] is a newly proposed architecture for the future Internet. It is categorized into Information-Centric Networking (ICN) that has been gaining attentions from research communities. We need such a new architecture since the way we use the Internet has been changed since the date it was built. The majority of traffic on the Internet comes from content retrieval, instead of end-host communications. The Internet users do not care where their desired content is, instead, they just want to receive their requested data as fast as possible.

The CCN architecture was adapted to the growing trend by decoupling the location of the content repository from the routing. Instead, CCN identifies a content object by a unique name that is used in the routing mechanism. In addition, CCN suggests that each router should have a cache. By identifying content objects from unique names and caching them in routers, CCN seamlessly enables an in-network caching capability. Even though CCN provides the versatile caching capability, it strongly needs a proper cache management to efficiently utilize the limited resources. The CCN architecture suggests caching all content objects that traverse a router, which is a universal caching strategy, so that the cached content objects will serve some future requests. By using the in-network caching, some content requests find their desired content objects in some intermediate routers. These requests accordingly travel in shorter distances to reach their target content objects. The in-network caching potentially reduces the round-trip time, network traffic, and server load, which are beneficial to the content requesters, Internet service providers, and content providers, respectively. However, there have been evidences pointing out the inefficiency of the universal caching strategy [2], [3], [4], [5], [6]. Apart from sophisticated caching schemes, we observe that randomly caching content objects at a certain probability, which is called a probabilistic caching scheme, can overcome some main drawbacks of universal caching strategy in content-centric networks. We are aware of its promising performance and recognize the feasibility of practical implementation.

The probabilistic caching scheme was used as a benchmark policy in the literature [2], [3], [4] despite its interesting performance gain. In addition, the source of determining its caching probability remains ambiguous. The insights into the behavior of the probabilistic caching scheme have never been studied. It has been roughly evaluated when it works with a particular cache replacement policy (i.e., Least-Recently-Used (LRU)). To the best of our knowledge, there has never been a criterion that suggests a decent value of the caching probability as well as its practical limitation. Even though there are other caching schemes that can efficiently utilize the network of caches [4], [7], [8], [9], [10], those sophisticated caching strategies in general add significant complexity and overheads to caching systems, which may not be applicable from the practical point of view. We, therefore, focus our study on the behavior of probabilistic caching scheme, which is simple but surprisingly effective, when it works with different cache replacement policies. The objective of this study is to provide guidelines for managing content-centric networks that are not only efficient but also practical.

In this paper we explore the potentials and limits of a probabilistic caching scheme in content-centric networks. Computer simulations are performed to study the behavior of the probabilistic caching scheme when it works with different cache replacement policies, i.e., Least-Recently-Used (LRU), Least-Frequency-Used (LFU), and Randomly Replace (RR). We conduct under various parameter settings the simulations on a cascading topology and a hybrid topology, which is based on a real Internet topology. Our simulation results show different, interesting behavioral characteristics of the probabilistic caching scheme as a function of a cache replacement policy. The probabilistic caching scheme gives the improvement in the server load, round-trip hop distance, and cache hit rate compared with a universal caching scheme only when it works with LRU. The improvement increases as an inverse function of the caching probability assigned to the probabilistic caching scheme. On the contrary, the probabilistic caching scheme does not work well when each router implements LFU. In addition, varying the caching probability of the probabilistic caching scheme does not impact the in-network caching performance when RR is deployed in content-centric networks. The initial state of a network of caches is long for a small caching probability of the probabilistic caching scheme regardless of the deployed cache replacement policy.

The rest of this paper is organized as follows. Section II presents the related work. Section III describes the CCN architecture and its relation to a network of caches. Section IV describes the caching system model and its functions used in this study. We include the definition of several caching and cache replacement policies as well as enumerating their important properties in the section. Section V presents the simulation results, discussion, and conclusive remarks. We conclude this work in Section VI.

II. Related Work

The evaluation of caching schemes and replacement policies has been extensively studied. However, caching and replacement methods, which cooperatively manage a caching system, have been frequently evaluated as separate studies. A number of caching strategies have been recently proposed [4], [7], [8], [9], [10], [11]. Content-popularity-based caching schemes were presented in [7], [8]. These novel caching schemes have been proven to be efficient but they inevitably require synchronization among nodes, which may introduce large overheads and complexity. A few cooperative caching strategies were presented in [4], [9], [10] but those sophisticated caching schemes may be impractical, considering the scale of content-centric networks as well as the link latency. Chai et al. [11] proposed a centrality-based caching algorithm whose caching decision is based on the concept of betweenness centrality. The performance of their caching method was supported by simulation results but it was evaluated only with LRU.

In previous studies of cache replacement policies, LRU and LFU were commonly considered, whereas other replacement policies were left out of the scope of interest [12], [13], [14]. Carofiglio et al. [12] explored the impact of storage management on the performance of multiple applications that concurrently share the same content-centric network. They observed that the performance of LFU is superior to that of LRU in terms of the diversity of content cached in a network. Ardelius et al. [13] provided analytical solutions for the cache hit rate and data availability of an aggregation access network. Nevertheless, only a universal caching scheme was considered in their analytical models which cannot explain the behavior of a probabilistic caching scheme. Fricker et al. [14] studied the impact of traffic mix on the caching performance in content-centric networks. They pointed out the inefficiency of LRU in comparison to LFU. However, their study was limited to a two-layer cache hierarchy, which may not be applicable to large networks of caches.

A more complete study that considered both caching schemes and cache replacement policies was conducted by Rossi and Rossini [3]. They evaluated the different combinations of caching schemes and cache replacement policies in various network topologies. They observed that the results of matching between randomly caching and random replacement policies were as good as that of the matching between a universal caching scheme and LRU. However, they did not clearly state the causes of their results, which is necessary for understanding the behavior of a probabilistic caching scheme. Therefore, we conduct our study with the goal to clarify the performance of the probabilistic caching scheme under various cache replacement policies.

III. Content-Centric Networking and Network of Caches

The Content-Centric Networking (CCN) architecture was proposed in [1]. CCN uses prefix names to identify the content objects and to route packets. The prefix name can be hierarchically constructed based on the URI Representation. There are three main data structures in a CCN router, i.e., Forwarding Information Base (FIB), Content Store (CS), and Pending Interest Table (PIT). The FIB of a CCN router partly resembles the routing table of IP routers whose destination field is changed into the prefix of the content names. The CS is a cache embedded in a CCN router. The PIT is responsible for keeping the routing states of ongoing transmissions. The routing in CCN is receiver-driven, which is controlled by the forwarding of *interest packets*. An interest packet contains a prefix name of the content. When an interest packet arrives at a CCN router, it searches for the desired content that may be stored in the CS. If there is a matching content, a *data packet* is created and sent back along the reverse path of the relevant interest packet. Otherwise, the CCN router checks the PIT whether any interest packet asking for the same content has been sent. If so, the ingress face (interface) of the interest packet is added to the existing PIT entry. Otherwise, a new PIT entry is created and the FIB is consulted to determine where to forward the interest packet. When the forwarded interest packet meets its desired content, the relevant data packet follows the reverse path to the requester router based on the recoded ingress faces in the PIT. If the prefix name of the data packet matches a PIT entry, it is considered valid. Otherwise, the data packet is invalid and should be discarded. The content in a valid data packet can be cached in the CS of each CCN router it traverses depending on the caching decision and cache replacement policies.

In general, a content-centric network consists of a number of CCN routers, which can be considered a network of caches. A traditional IP network that deploys caching systems, e.g., web caches and Content Distribution Networks (CDNs), is also a network of caches. However, the content-centric and IP networks are different in terms of their caching granularity. An IP network may have several caches or data centers that are monitored by an administrator. In the case of CDN, the content stored in the data centers are deliberately selected in advance by content providers. On the other hand, the number of CCN routers in a content-centric network, which is seen as the number of independent caches, can vary from a few to several thousand nodes depending on where the CCN architecture is deployed. A large number of nodes in a content-centric network as well as the link latency between them could make the cooperative caching and centralized management infeasible or ineffective.

IV. Caching System Model

There are two important algorithms managing a caching system: a caching scheme and a cache replacement policy. The capacity of a cache is limited by the embedded physical memory whose size is smaller than the item population. A *caching scheme* decides whether a caching system stores an item in its cache. An empty cache becomes full as a result of storing such items. A *cache replacement policy* is then needed when a new item enters the cache system and requires some space. A currently cached item is selected to be a victim for an eviction. The victim selection is governed by the cache replacement policy. The objective of a sophisticated cache replacement policy is to keep the items that are likely to be requested in the future by replacing an item that tends to be useless for future requests. A commonly used criterion for evaluating a caching system is its *hit ratio*, i.e., the frequency that a request finds its desired item in the cache. The caching schemes and cache replacement polices used in this study as well as their properties will be stated as follows.

A. Caching Schemes

The routers in a network should individually perform in a distributed manner and operate fast in order to fully exploit the benefit of CCN. Otherwise, the CCN architecture itself may cause a bottleneck in the network. We, therefore, focus on the two most straightforward and commonly known caching schemes, the universal and probabilistic caching schemes, which are considerably practical and scalable to a wide range of network sizes.

1) Universal Caching Scheme (Always): The default CCN architecture suggests that a router should exploit a universal caching strategy, which is referred to as $Always$ hereafter, as a caching decision policy of each CCN router [1]. A CCN router that deploys $Always$ as its caching policy always caches the content object extracted from a valid data packet. The approach can quickly distribute content in a content-centric network. However, there are evidences pointing out that $Always$ can put the replicas of the same content objects in multiple CCN routers and thus degrades the overall performance of in-network caching, which is indicated by low cache hit rates at intermediate routers [2]. However, the poor performance of *Always* has been confirmed when it is merely used with a particular cache replacement scheme, i.e., Least Recently Used (LRU). Therefore, we further investigate the behavior of *Always* when it is used with other replacement schemes in the next section.

2) Probabilistic Caching Scheme (Prob(p)): The probabilistic caching scheme, which is referred to as $Prob(p)$ from now, was used as a benchmark scheme in the literature [2], [3], [4]. The key idea is that each CCN router randomly caches a content object that traverses it at a certain caching probability, which is defined by p, where $0 < p < 1$. $Always$ is a special case of $Prob(p)$, where $p = 1$. To the best of our knowledge, there has never been an unambiguous criterion that suggests a decent value of p and we first focus on this issue. Interestingly, the performance of $Prob(p)$, where $p < 1$, is better than that of $Always$, which can be inferred from the improvement in server hit rate and hop distance [2], [4], [7]. However, the $Prob(p)$ has been tested only with a cache replacement policy, LRU. Note that $p = 0.1$ is the lowest value of p that has ever been used [4]. We propose that the value of p could be further decreased to improve the caching performance while its limit is constrained by an acceptable duration of the transit state of caching systems. The important properties of the $Prob(p)$ can be summarized as follows: 1) Decreasing the caching probability p in $Prob(p)$ reduces the probability that multiple CCN routers on a delivery path cache the same content object in a content delivery; and 2) Decreasing the caching probability p of $Prob(p)$ results in a longer duration of the initial state of caching systems, given a static request pattern.

The first property suggests that a small value of p should be assigned to $Prob(p)$ in order to effectively distribute multiple content objects in a content-centric network and to efficiently utilize the in-network caching ability of CCN. In other words, the diversity of the content objects cached in the network can be improved by decreasing the value of p. However, setting a small value of p may result in a long duration of the initial state of caching systems according to the second property, which leads to a poor performance of capturing a high variation of access patterns.

B. Cache Replacement Policies

The capacity of a cache is generally smaller than the population of items, so all of such items cannot simultaneously reside in the cache. If the cache is full, the caching system must discard one of currently cached items before it can store a new item. A cache replacement policy determines which item is evicted. We consider three commonly known algorithms: Least Recently Used (LFU), Least Frequency used (LRU), and Randomly Replace (RR).

LRU tries to keep recently active items in the cache by discarding the item that is least-recently-used. LRU is simple to implement and operates fast since its running-time per request is $O(1)$. However, if the capacity of cache is not large enough, LRU poorly performs when items are requested in a round robin fashion. Items will consistently enter and leave the cache without cache hit occurs. LFU replaces the least-frequently-used item with a new one. LFU is optimal when the requests received at different times are stochastically independent [15]. However, the running-time per request is logarithmic in the cache size ($O(\log(n))$), where n is the cache size. In addition, it adapts poorly to variable access patterns by accumulating stale items with past high-frequency counts. RR is the simplest replacement policy—one of currently cached items is randomly evicted whenever a replacement is invoked. It does not keep past information of access patterns and thus requires the minimal system requirements to operate. We use RR as a reference for the aforementioned policies, i.e., LRU and LFU. To the best of our knowledge, most of previous studies used LRU and RR as replacement policies of CSs in content-centric networks when $Prob(p)$ was deployed [2], [3], [7]. We, therefore, evaluate our caching schemes of interest, i.e., $Always$ and $Prob(p)$, when they work with LRU, LFU, and RR by means of simulation in the next section.

V. Simulation Results

A. Simulation Set-up

We use ndnSIM [16], which is a NS-3 based network simulator dedicated to named data networking study, to conduct our simulations. All basic structures of CCN, FIB, PIT, and CS, are reproduced by ndnSIM. For the time being, ndnSIM models the routing mechanism of CCN which is driven by exchanging an interest packet and a data packet. However, it does not allow variable size of content object, so we ignore the content segmentation in our simulations and assume an identical size of content objects.

We assign various values of the caching probability p to $Prob(p)$, where $p \in \{1.0, 0.7, 0.3, 0.01\}$. As a result, our simulations take into account $Always$, $Prob(0.7)$, $Prob(0.3)$, and $Prob(0.01)$. We use LRU, LFU, and RR as a replacement policy of the CS of each CCN router. We vary the CS size of each node from 1% to 10% of the content population. We use the Dijkstra's algorithm to calculate the shortest path (in terms of the number of hops) to reach every content provider. Then we translate the calculated path to the FIB of each node.

The profile of content requests is modelled by using the Zipf's distribution which describes the popularity of each content object. Let M denote content catalog cardinality and $1 \leq i \leq M$. The probability of requesting a content with rank i is $\frac{1}{C \times i^{\alpha}}$ with $C = \sum_{j=1}^{M} 1/j^{\alpha}$, where α is an exponent parameter that shapes the content requests. A decent value of α has been being ambiguous in recent studies focusing on the caching performance of content-centric networks [3], [11], [14]. The value of α directly depends on the types of particular contents [14]. Equally important, different values of α can be derived from different geographical locations [17]. We, therefore, use $\alpha = 1.0$ as it was used in [11]. From our simulation results, the order of the performance among the caching and cache replacement policies are unchanged for $0.4 \leq \alpha \leq 1.6$, so the results presented in this paper are valid for these values of α.

Each simulation run begins with all CSs being empty (i.e., cold start). Unless otherwise specified, the simulations run with the following parameters. The total simulation time is equal to 10,000 seconds with 4,000 second warm-up period. Each content requester requests content objects following the Poisson process whose mean is equal to 50 requests/s. Each content provider serves 1,000 different content objects. The uniform size of CS is varied from 1%, 2%, 5%, and 10% of the total content population. The results are reported at 95% confidence interval.

B. Evaluation Metrics

We evaluate the performance of each caching scheme when it works with the particular cache replacement policy by observing four metrics: the server load, round-trip hop distance, hit rate, and instantaneous behavior.

Server load represents a prospective benefit of using CCN from the content provider standpoint. It directly reports the volume of traffic that a server must generate in response to its received requests. At an absence of cache in a network, the server load is equal to aggregated requests from all content requesters. An increasing server load pushes content providers to upgrade their facilities to provide an acceptable quality of service to all content requesters.

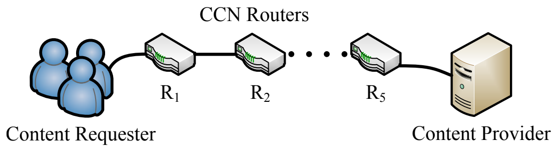


Fig. 1: Cascading network used in our simulations.

Round-trip hop distance is the sum of hops used by an interest packet and corresponding data packet in a trip of content retrieval. The round-trip hop distance is shorter if a requester can find its desired content object in nearby CCN routers. It also implies the **access latency** and partially estimates the **network load**.

Cache hit rate is commonly used to evaluate caching system. A high hit rate of a caching system implies its good performance. The hit rate of a CCN router is the probability that a content request finds its desired content objects in the CS of a CCN router. However, the hit rates of different routers may differently contribute to the overall performance of a content-centric network.

Instantaneous behavior is observed to illustrate how fast or slow a caching scheme can adapt caching systems to an access pattern. We can observe the duration of the initial state of a network of caches from the instantaneous behaviors of server load, round-trip hop distance, or hit rate. Due to the limited space, we show only the instantaneous behavior of the server load in this paper.

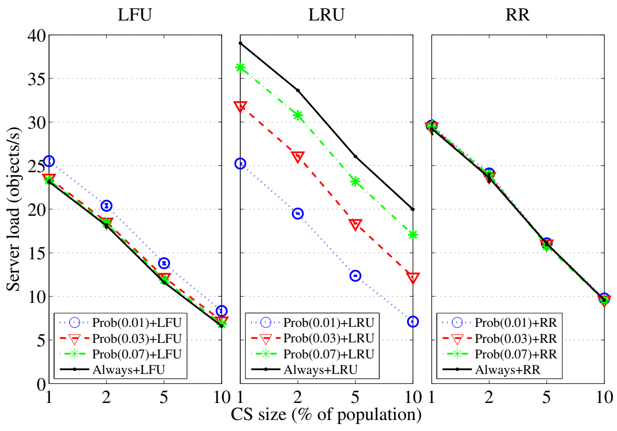
C. Network Topologies

We conduct our simulations on two network topologies, which are described as follows.

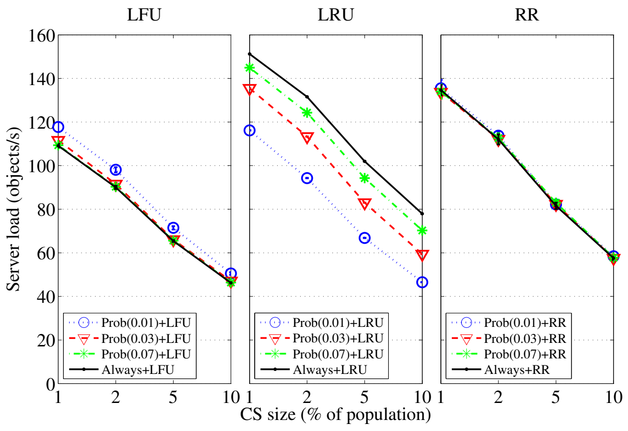
1) Cascading Network: We use a fixed length cascading network in our simulations. A cascading network contains five CCN routers as shown in Fig. 1. We consider two study cases of using the cascading network in order to cover its practical use.

- **One content requester**: The first study case is that request traffic accesses the cascading network at one CCN router. A content requester is connected to the CCN router at one end of network (R_1), whereas a corresponding content provider is connected to the CCN router at the other end (R_5).
- **Multiple content requesters**: The second study case considers that requests enter the network through multiple CCN routers. More specifically, four content requesters are connected to routers R_1, R_2, R_3, and R_4. These content requesters request content objects from the content provider that is connected to the end of the network (R_5).

2) Real Internet Topology SINET4: Another network topology of interest is based on the Science Information NETwork 4 (SINET4) [18]. SINET4 is providing a unified network connection to 700 universities and research institutes in Japan. SINET4 is constituted of eight core nodes and 42 edge nodes that are geographically distributed around Japan. Some routers are connected to foreign academic networks and commercial

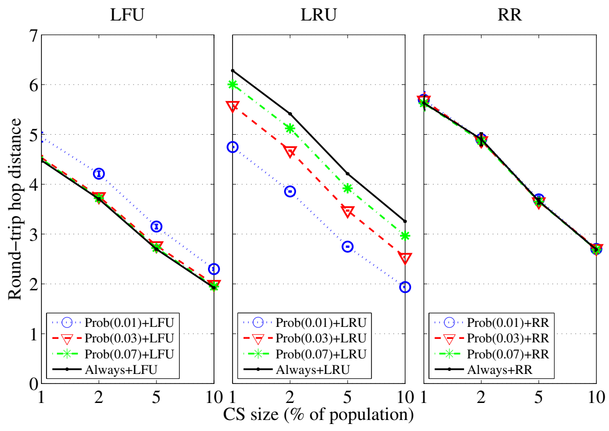


(a) One content requester

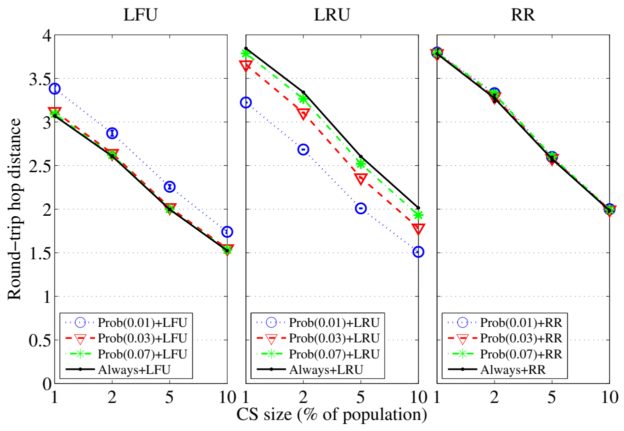


(b) Multiple content requesters

Fig. 2: The server load for different CS sizes in the cascading network.



(a) One content requester



(b) Multiple content requesters

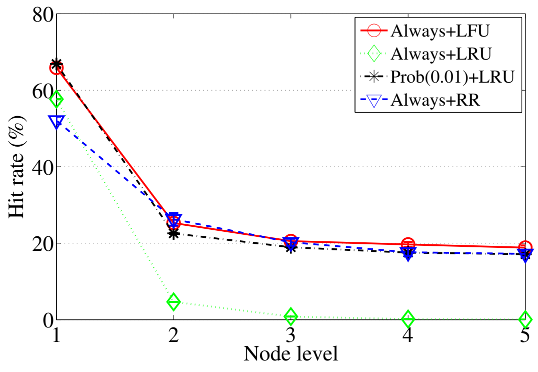
Fig. 3: The round-trip hop distance for different CS sizes in the cascading network.

Internet service providers to link SINET4 to the globe. SINET4 forms a hybrid topology that consists of a mesh network and a number of star topology networks. The eight core nodes constitute a mesh network, whereas each core node is connected to a number of edge nodes to form a star topology network. We conduct our simulations using the network topology of SINET4 as follows. All nodes in the network represents the identical CCN routers. We link a unique content provider to each core node. Each content provider provides 1,000 unique content objects so that 8,000 content objects are available in total. A content requester is connected to each edge node so that there are 42 content requesters in the network. Each content requester equally requests content objects from all content providers.

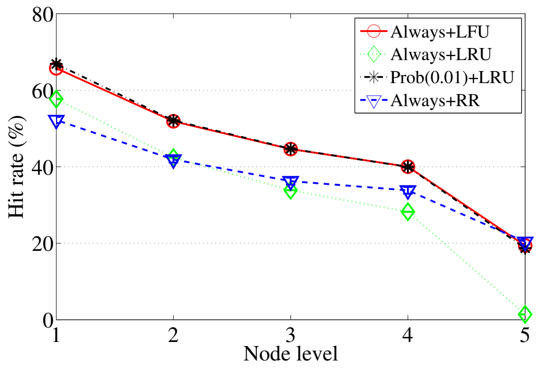
D. Results and Discussions

1) Experiments with Cascading Network: We report the average server load of the content provider for each caching scheme that works with different cache replacement policies in Fig. 2. The results for the first and the second study cases of a cascading network are shown in Figs 2 (a) and (b), respectively. We find that the performance of server load reduction varies as functions of the caching scheme, cache replacement policy, and the size of CS. We observe that the server load decreases as a function of CS size for all cases. Surprisingly, each replacement policy shows its unique behavior given the different caching schemes. $Prob(p) + LRU$ gradually reduces the server load when the value of p is decreased. Specifically, $Prob(0.01)+LRU$ yields 20% improvement in the server load reduction over $Always+LRU$. The improvement comes from the reduced replicas of the same content in multiple routers. These results also apply to the second study case.

In contrast, we obtain the opposite results for $Prob(p) + LFU$ and $Always + LFU$. $Prob(p) + LFU$ increases the server load when the value of p decreases. This is due to the poor performance of LFU against variable access patterns. Specifically, each CCN router may accumulate stale items with past high-frequency counts. For instance, a popular content object may be cached by a CCN router at the initial state of the caching system as a result of $Prob(p)$, where $0 < p < 1$. The cached content object then starts to accumulate the reference frequency and continues to reside in the CCN router if it is popular at that time. The accumulated reference frequency of the content object is a nondecreasing function as long as the content object can continuously stay in the CCN router. When the cache enters its steady state, this content object may become less popular than other content objects from the router's standpoint. This is because the interest packets corresponding to this content object are already satisfied by another router nearby the content requester. However, the

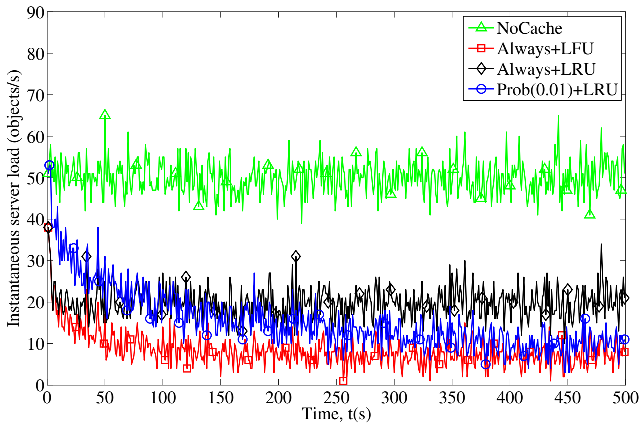


(a) One content requester

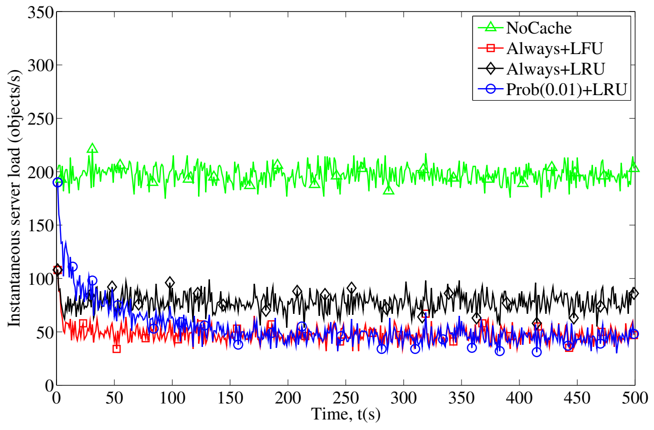


(b) Multiple content requesters

Fig. 4: Hit rate with respect to the node level in the cascading network.



(a) One content requester



(b) Multiple content requesters

Fig. 5: Instantaneous behavior of different caching schemes and replacement policies in the cascading network.

accumulated reference frequency of the content object may be higher than that of currently popular content objects. As a result, the stale content object pollutes the cache when LFU is used. In addition, $Prob(p)$ intensifies the issue when the value of p is smaller, since the stale content object may stay in the cache for a longer period. This comes from the fact that the CCN router randomly caches fewer incoming content objects due to a lower caching probability.

We find that $Prob(p) + RR$ does not yield any significant change to the results in terms of the server load reduction, although we alter the values of p. The results for $Always+RR$ and $Prob(p) + RR$ are almost identical. It is because RR naturally distributes content objects in the network regardless of the caching scheme used in our simulations. The above statement applies to the results of the second case study (i.e., content requests enter the network through multiple routers) as shown in Fig. 2. It is worth to note that the performance of $Always+LRU$ regarding the server load reduction is inferior to that of $Always+RR$ in both study cases. However, we find that the gap between the performance capabilities of different caching schemes and cache replacement polices for the second study case is smaller than that of the first case.

In essence, $Always+LFU$, gives the best performance as a reward for its complexity. On the contrary, $Always + LRU$ performs the worst in all cases. The performance of $Prob(p)$ obviously depends on the value of p when it works with LRU. This is because LRU tends to leave the same content object in neighbor nodes, and $Prob(p)$ with a smaller value of p prevents this from happening.

The results in terms of round-trip hop distance are shown in Figs. 3 (a) and (b) for the first and second study cases, respectively. The order of performance among the caching and cache replacement policies in terms of round-trip hop distance is similar to that in terms of server load. On the other hand, the performance of $Always+LRU$ in terms of round-trip hop distance is significantly inferior to that of $Always + RR$ in the first study case, whereas it is almost identical to that of $Always + RR$ in the second case. The results come from the fact that the performance of $Always+LRU$ highly depends on where the requests enter the cascading network, which can be observed from the results in terms of cache hit rate as follows.

We measure the cache hit rate of CCN routers towards the node levels and report them in Figs 4 (a) and (b) for the first and second study cases, respectively. We show a comparison of $Always + LFU$, $Always + LRU$, $Prob(0.01) + LRU$, and $Always + RR$ when the CS size of each router is equal to 10% of population due to limited space. For the first study case where the requests access the network at the node level 1, $Always + LFU$ gives the best hit rates for all node levels in comparison to the other schemes. Interestingly, $Prob(0.01) + LRU$ remarkably overcomes $Always + LRU$ for all node levels. We find that $Always + LRU$ gives a high

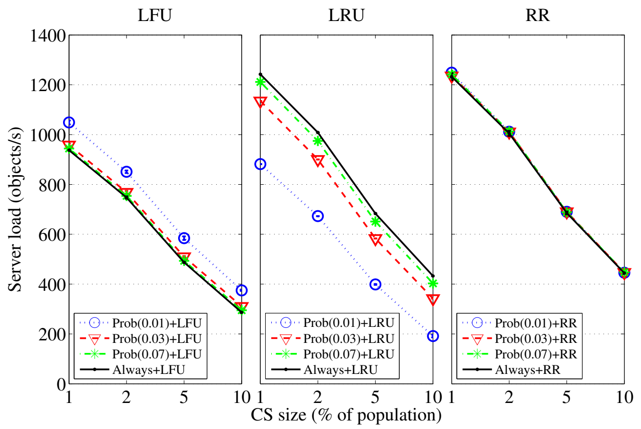


Fig. 6: The server load for different CS sizes in SINET4.

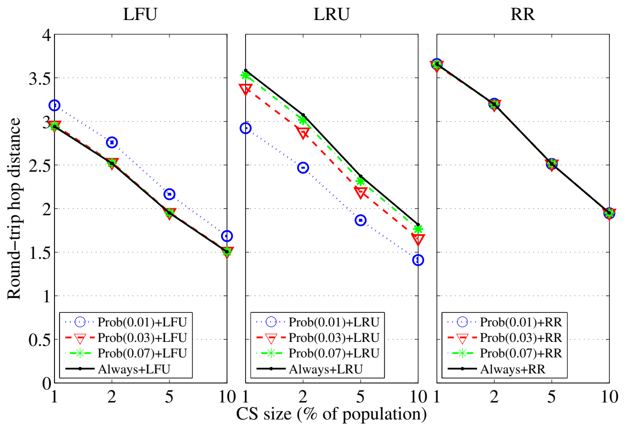


Fig. 7: The round-trip hop distance for different CS sizes in SINET4.

hit rate for the node level 1 whereas the other nodes all suffer limited hit rates. In fact, $Always + RR$ achieves higher hit rate than $Always + LRU$ for every node level except the node level 1. For the second study case, the requests enter the network through multiple routers, so these routers become the first hop routers of some requests. $Always+LRU$, in essence, performs well for the first hop router, so its performance for the second case is better than that for the first case. The hit rates of all node levels for $Always + LFU$ and $Prob(0.01) + LRU$ are almost identical. $Prob(0.01) + LRU$ is easier to implement than $Always+LFU$, considering their running-time per request and caching-cost per transmission. In other words, although $Prob(0.01)$ lets each router cache fewer content objects than $Always$ in a transmission, it still gives the comparable performance.

We next show in Figs. 5 (a) and (b) the instantaneous behaviors of the different caching schemes and cache replacement policies in comparison to the network without cache ($NoCache$). We show only the tracks of the server load for $Always + LFU$, $Always + LRU$, and $Prob(0.01) + LRU$ when the CS size is equal to 10% of the population, since the order of the server loads provided by different cache management schemes is unchanged for all considered CS sizes. We do not show the results for $Prob(0.01) + LFU$ since they are almost indistinguishable from that of $Prob(0.01) + LRU$. We find that both $Always + LFU$ and $Prob(0.01) + LRU$ perform better than $Always + LRU$ in their steady states for both study cases. However, we observe that $Prob(0.01)+LRU$ takes much longer duration of the initial state than the others.

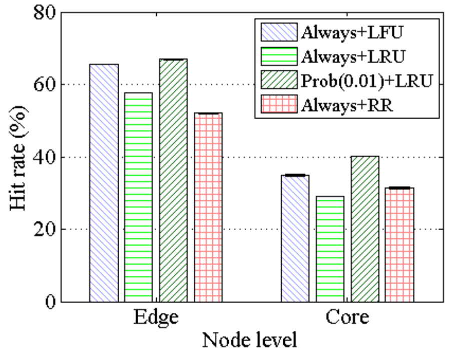


Fig. 8: Hit rate towards the node level in SINET4.

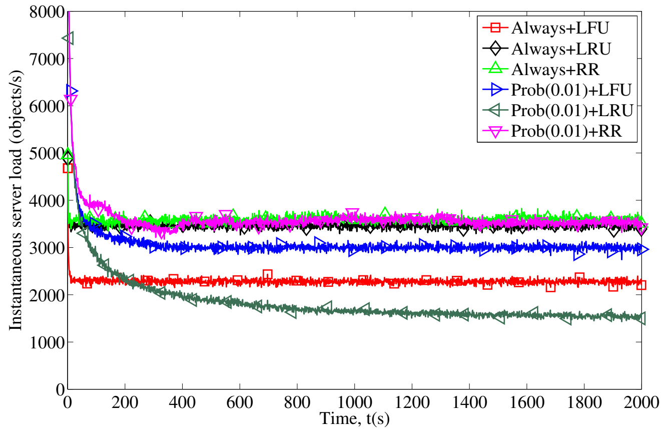


Fig. 9: Instantaneous behavior of different caching schemes and replacement policies in SINET4.

The long initial state of a network of caches caused by $Prob(0.01) + LRU$ may not be suitable to a request pattern with a high variation.

2) Experiments with Real Internet Topology: We report the results for SINET4 topology in terms of the server load reduction, round-trip hop distance, hit rate, and instantaneous behavior in Figs. 6, 7, 8, and 9, respectively. Fig. 6 shows the average server load for all content providers that are linked to the network. The results partly resemble that of the cascading network in the previous section. $Prob(p) + LRU$ significantly improves the server load reduction when the value of p is decreased. The improvement is a result of the probabilistic caching scheme that minimizes the probability that multiple CCN routers cache the same content object. In addition, $Prob(p)$ alleviates the downside of LRU that occurs when request patterns follow a round-robin fashion. $Prob(p)$ allows a cached content object to longer stay in a CCN router until the router caches a new content object. In contrast, we observe that $Prob(p) + LFU$ performs worse than $Always + LFU$ when $p \in \{0.01, 0.3, 0.7\}$ which is reflected by the increased server load. Thus, we infer from the results that $Always$ is the best caching scheme given the LFU as a replacement policy. The results show the incompatibility of using $Prob(p)$ with LFU and reiterate the importance of matching the particular caching scheme to a cache replacement policy. Interestingly, $Prob(0.01) + LRU$ even overcomes $Always + LFU$ in the SINET4 topology. It is because LFU suffers from accumulating the content objects with past high-frequency counts whereas LRU can benefit from the content objects distributed in the network. As we expect, $Always + RR$ and the variants of $Prob(p) + RR$ all give indistinguishable results.

We report in Fig. 7 the round-trip hop distance for different caching schemes and replacement policies at various sizes of CS. The results in terms of the round-trip hop distance follow the trend of server load for all cases. We observe that $Prob(0.01) + LRU$ even gives a better round-trip hop distance than that of $Always + LFU$ for a few percentages. RR consistently yields the worst round-trip hop distance among those of the other replacement policies regardless of the exploited caching policies.

We next show in Fig. 8 the average hit rate of CCN routers whose CS sizes are equal to 10% of the content population as a function of the node levels, i.e., core and edge nodes. We find that $Prob(0.01) + LRU$ gives better hit rates than $Always + LFU$ and the other schemes. $Prob(0.01) + LRU$ significantly improves the hit rates of both core and edge nodes (up to 10%) over that of $Always + LRU$.

Figure 9 shows the instantaneous behaviours of $Always$ and $Prob(0.01)$ when they work with LFU, LRU, and RR in the SINET4 topology where the CS size of each router is equal to 10% of content population. The results show that the initial state of the caching system for $Prob(0.01) + LRU$ is longer than those for $Always{+}LFU$, $Always{+}LRU$, and $Always{+}RR$. The instantaneous behaviors of $Prob(0.01) + LFU$ and $Prob(0.01) + RR$ also follow that of $Prob(0.01) + LRU$. Nevertheless, $Prob(0.01) + RR$ results in a shorter duration of the initial state of caching systems than $Prob(0.01) + LFU$ and $Prob(0.01) + LRU$. However, it gives the poorest caching performance in the steady state of the caching systems compared with the others.

3) Conclusive Remarks: The behavior of a probabilistic caching scheme explicitly varies as a function of a cache replacement policy. The probabilistic caching scheme gives the improvement in the server load, round-trip hop distance, and cache hit rate compared with a universal caching scheme only when it works with LRU. The improvement increases as an inverse function of the caching probability assigned to the probabilistic caching scheme. When LFU is deployed in a content-centric network, a universal caching scheme is a policy of choice since it gives a better performance than the probabilistic caching scheme. On the contrary, the probabilistic caching scheme even magnifies the issue of LFU by letting CCN routers accumulate stale content objects with past high-frequency counts. The probabilistic and universal caching schemes have an identical behavior when RR is deployed in content-centric networks. The initial state of a network of caches is longer when the caching probability of a probabilistic caching scheme is decreased regardless of the deployed cache replacement policy.

VI. Conclusion

We study the behavioral characteristics of a probabilistic caching scheme by means of computer simulation. We evaluate the probabilistic caching scheme when it works with different cache replacement policies. The evaluation metrics consist of the server load, round-trip hop distance, cache hit rate, and instantaneous behaviour. The simulation results show that the behavior of a probabilistic caching scheme explicitly varies as a function of a cache replacement policy. The performance of probabilistic caching scheme and the duration of the initial state of a network of caches are inverse functions of a caching probability. The probabilistic caching scheme works well only in the caching system that implements Least-Recently-Used (LRU) as a cache replacement policy, whereas the limit of its performance comes from the increased duration of the initial state of the caching system.

Acknowledgment

This research is supported in part by JSPS Grants-in-Aid for Scientific Research A (23240011).

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, pp.1-12, 2009.

[2] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. of the second edition of the ICN workshop on Information-centric networking*, pp.55-60, 2012.

[3] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Tech. Rep., Telecom ParisTech*, 2011.

[4] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. of IEEE INFOCOM WKSHPS 2012*, pp.316-321, March 2012.

[5] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the scale and performance of coopeartive Web procy caching," *Operating Systems Review*, vol.34, no.5, pp.1631, Dec. 1999.

[6] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proc. of ACM WKSHPS HotNets-X*, pp.1-6, 2011.

[7] H. Wu, J. Li, T. Pan, and B. Liu, "A novel caching scheme for the backbone of named data networking," in *Proc. of IEEE ICC 2013*, pp.3634-3638, June 2013.

[8] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *Proc. of IEEE ICC 2013*, pp.3619-3623, June 2013.

[9] S. Saha, A. Lukyanenko, and A. Yla-Jaaski, "Cooperative caching through routing control in information-centric networks," in *Proc. of IEEE INFOCOM 2013*, pp.100-104, 2013.

[10] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS cooperative caching for content-centric networks," in *Proc. of ACM SIGCOMM WKSHPS ICN 2013)*, pp.61-66, 2013.

[11] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *Proc. of the 11th international IFIP TC 6 conference on Networking (IFIP'12)*, pp.27-40, 2012.

[12] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in Content-Centric Networking," in *Proc. of IEEE ICC 2011*, pp.1-6, June 2011.

[13] J. Ardelius, B. Gronvall, L. Westberg, and A. Arvidsson, "On the effects of caching in access aggregation networks," in *Proc. of ICN WKSHPS on Information-centric networking*, pp.67-72, 2012.

[14] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proc. of IEEE INFOCOM WKSHPS 2012*, pp.310-315, March 2012.

[15] J. E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall, 1973, pp.282.

[16] A. Afanasyev, I. Moiseenko, and L. Zhang, ndnSIM: NDN simulator for NS-3," *Tech. Rep. NDN-0005, University of California, Los Angeles*, 2012.

[17] A. Brodersen, S. Scellato, and M. Wattenhofer, "Youtube around the world: geographic popularity of videos," in *Proc. of 21st International World Wide Web Conference*, pp.241-250, 2012.

[18] SINET4: Science Information NETwork 4 [Online]. Avalible: http://www.sinet.ad.jp.