International Journal of Innovative Computing, Information and Control
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **519–535**

AN EFFICIENT RANKING-MATCHED CACHING STRATEGY FOR INFORMATION CENTRIC NETWORKING

Li Ding[1], Jinlin Wang[2], Yiqiang Sheng[2] and Lingfang Wang[2]

[1]Department of Automation
University of Science and Technology of China
No. 96, Jinzhai Road, Baohe District, Hefei 230026, P. R. China
dingli@mail.ustc.edu.cn

[2]National Network New Media Engineering Research Center
Institute of Acoustics, Chinese Academy of Sciences
No. 21, North 4th Ring Road, Haidian District, Beijing 100190, P. R. China
{ wangjl; shengyq; wanglf }@dsp.ac.cn

Received August 2017; revised December 2017

Abstract. *The host-based communication model of current Internet infrastructure is proved to be inefficient in content distribution because of a lot of bandwidth waste. In this context, Information Centric Networking (ICN) emerges as a promising future architecture that aims to distribute content efficiently over the network. The key to efficient content distribution in ICN is in-network caching, which is an effective way of eliminating redundant network traffic. Cache decision at each router is a very important factor which governs caching performance. In this paper, we propose a content Popularity Ranking and node Importance Ranking Matched (PRIRM) caching strategy for ICN, which is targeted at reducing cache redundancy and improving content diversity. Content popularity represents the user preference while node importance determines the important degree in the network. By disseminating content on the delivery path based on the popularity ranking of the content and the importance ranking of the node, content with different rankings can be distributed in the network hierarchically. The sensitivity of PRIRM strategy against different parameters, such as content number, popularity distribution and network topologies, is thoroughly studied. The simulation results show that PRIRM strategy can result in a significant decrease of average hop count, compared with three benchmark caching strategies, i.e., leave copy everything strategy, a centrality-based strategy and a path-capacity-based strategy.*

Keywords: In-network caching, Content popularity, Betweenness centrality, Information centric networking

1. **Introduction.** Today the Internet is a complex and heavily loaded multimedia/information system based on content distribution [1]. Content services are experiencing rapid growth, especially the video streaming services such as YouTube. Current host-to-host communication paradigm is inefficient in content distribution due to a lot of bandwidth waste [2]. Users are more interested in services, rather than sources. Many researchers are committed to solving the low efficiency of content distribution and new architectures have been proposed, which focus on the content rather than the location [3, 4]. As one of the promising future Internet architectures, Information Centric Networking (ICN) [5] has attracted considerable attention in academia. One of the crucial characteristics of ICN is the ubiquitous in-network caching, which means that a router can cache previously forwarded data packets and potentially serve for subsequent requests. Consequently, transfers of redundant traffic can be reduced and Quality of Service (QoS) can be improved.

ICN brings about both opportunities and challenges for researchers [6]. The research about in-network caching of ICN focuses on two aspects: caching decision policy and caching replacement strategy [7]. The former is the most relevant to our work. Caching decision policy determines which objects are to be placed at which caching nodes. It can be classified into two groups further: on-path caching and off-path caching. In on-path caching, the requested content can be only stored at the appropriate node on the delivery path. When a router is not located on the delivery path, it will not cache any packets. The disadvantage is that on-path caching cannot make full use of in-network nodes. On the contrary, in the off-path caching, cache replicas can be located at a wide scope rather than be confined to the delivery path [8]. For example, over the last years the ICN research community has proposed hash-routing schemes [9, 10]. According to these schemes, edge nodes will implement a hash function mapping the content identifier to a specific caching node. When a content is on the reverse path to the requester, edge nodes can forward it to the caching node calculated by the hash function. As a result, requests can be routed to the corresponding caching nodes directly to attain a high cache hit. However, off-path caching usually requires some co-ordinations among caching nodes to obtain the optimal performance.

In this paper, we focus on on-path caching and are aiming at finding an efficient solution to reduce cache redundancy and improve content diversity. Leave Copy Everywhere (LCE) strategy [11] is the most simple content placement strategy. Nevertheless, this strategy will cause a high degree of redundancy as all caches along the delivery path consume cache resources to hold identical items. Meanwhile, since each router caches data packets without considering the popularity, popular content cannot stay long in a router because of a high replacement rate. Betweenness Centrality (BC) based strategy [12] argues that cache operation should only be performed at the most important node. Since the importance of each node can be measured by the betweenness centrality, the node with the highest betweenness centrality on the delivery path will cache the data packet. However, cache capacity is much smaller compared with the tremendous amount of content in the network [13]. Replacement on the most important node will take place frequently. Consequently, the most popular content has a high possibility of being replaced quickly. Thus, subsequent requests cannot be satisfied by previous cache. [14] proposes that nodes closer to users should cache popular data packets with high probability while unpopular contents can be stored closer to the server. This caching scheme can improve content diversity, but it ignores the fact that the nodes closer to users may not have enough importance in the network.

Considering the shortcomings of these methods, we propose a new caching strategy based on popularity ranking and node betweenness centrality ranking, which takes full account of the importance of the node and the popularity of the content. When a data packet is on the reverse path from the source to the consumer, each node will make a decision to cache it or not based on its betweenness centrality ranking and the popularity ranking of the content. This strategy can ensure that contents with different popularity rankings can be distributed on nodes with different betweenness centrality rankings correspondingly. In this way, the replacement rate of the most important node can be reduced and the content diversity can be improved.

The contribution of this paper can be summarized as follows:

1) We propose a content Popularity Ranking and node Importance Ranking Matched (PRIRM) caching scheme for information centric networking to reduce cache redundancy and improve content diversity.

2) We propose a time window based method to estimate content popularity and the total distinct content population. The estimated method can achieve a considerable performance as the global popularity in PRIRM strategy.
3) We make extensive experiments on ndnSIM [15] to evaluate the performance of PRIRM strategy. The sensitivity of this strategy against different parameters is thoroughly studied.

The remainder of this paper is structured as follows. Section 2 summarizes the related work. In Section 3, we propose our novel PRIRM caching decision policy. Section 4 presents our simulation results on ndnSIM. Finally, we make a conclusion and describe our future work in Section 5.

2. **Related Work.** The general problem of data caching has been widely investigated before in a variety of computer systems (applications) such as CPUs, storage systems and databases. For example, a memory system in computer system is a hierarchy of storage devices with different capacities, costs and access times. In Web caching system, there are usually multiple level caches. With hierarchical caching, the retrieval latency for Web documents can be reduced significantly [16]. However, efficient caching in ICN is nevertheless a great challenge because a single device should be able to perform wire-speed forwarding and caching at the same time. Hence, a caching scheme is required to be simple but effective [17]. In recent years researchers have proposed many interesting approaches to improving cache performance considering the low efficiency of the default caching strategy LCE. In general, these approaches can be classified into two categories, implicit coordination and explicit coordination [18]. In implicit coordination, each cache is managed in a distributed fashion and independently controlled. On the contrary, each node will work collaboratively with other nodes to make cache decision in explicit coordination. In this section, we highlight existing literature studies most related to our work.

Leave copy down strategy [19, 20] makes frequently requested contents closer to the user as a data packet is stored only in the next hop of cache hit on the path towards the client. A probability-based caching scheme is devised in [21, 22]. In these two approaches, a router will cache a data packet with fixed or variable probability to reduce redundancy. However, they do not consider the popularity of content, which means that arrival of unpopular content might cause the eviction of popular content. [23] points out that popularity is an important factor, affecting the performance of a caching algorithm. Hence, some researchers suggest that a router should consider the popularity of content when making cache decision [14, 24, 25, 26]. Specifically, [24] argues that each node should only cache popular data packets. A popularity table is maintained to determine whether a content is popular or not. [14] can expand popular content widely in the network and popular content is pushed to nodes close to users. [25] proposes two popularity-based progressive caching schemes which aim at populating edge routers with popular content and reducing the redundancy of cached items along the delivery path. [26] designs a collaborative caching strategy based on the local popularity statistic results. The core of this algorithm is to place a data packet on one node from the access routers to the gateways according to its popularity ranking. However, these approaches do not take account of the importance difference of caching nodes. [12] proposes a method to evaluate the importance of a caching node by its betweenness centrality. Since betweenness centrality measures the extent to which a vertex lies on paths between other vertices, a node with high betweenness centrality will have considerable influence within a network. Hence, content caching at these nodes can bring the maximum benefit. However, it fails to consider content

popularity. It is expected to design a caching strategy which considers popularity of content and importance of routers simultaneously.

In summary, a caching strategy considering only a single factor may not be effective enough in respect of reducing cache redundancy and improving content diversity. Hence, a combination of node betweenness centrality ranking and content popularity ranking is investigated in this paper and expected to have a better performance. Specifically, a data packet will be stored on the router according to the popularity ranking of the content and betweenness centrality ranking of the router on the delivery path. This strategy can avoid caching duplicate contents many times and distribute contents in the network hierarchically.

3. PRIRM Caching Strategy.

3.1. Motivation.

In the ICN architecture, a content store (CS) will be used as the buffer to cache content packets. Given that the size of CS is far less than the huge amount of content, it is expected to cache only a few popular content packets since caching popular data packets can bring about more benefits. Moreover, since the importance of network nodes is different, popular objects should be stored on important nodes as more requests will pass through these nodes on their forwarding paths. On the contrary, even the least important node stores the most popular content, there is a high chance that it will not be on the routing path of an interest packet. As a result, caching on this node still cannot be fully utilized. We envisage that contents can be distributed based on their popularity and the importance of network nodes. The most important node can store contents with higher popularity rankings while the least important node can hold the contents with lower popularity rankings on the delivery path. By deploying a hierarchical ranking-matched cache, cache redundancy can be reduced and content diversity can be improved.

3.2. PRIRM strategy.

3.2.1. *Ranking-matched scheme.* When a request is satisfied in the network, a data packet will be returned by taking the reverse path of the interest packet. The key question is how to select a proper node to reduce cache redundancy and improve content diversity. In order to illustrate our PRIRM strategy, we assume that the popularity ranking of a data packet, denoted by c, is $r(c)$. For the router x on the delivery path, its importance ranking among all nodes on the delivery path is $r(x)$. The cache probability for the data packet c that will be stored on the node x is denoted by $P_x(c)$. Let N and M represent the total number of caching node on the delivery path and the total diverse content population, respectively.

We divide the distinct content population into $\lfloor \frac{M}{N} \rfloor$ categories and let each router on the delivery path cache a certain category. Intuitively, a router with higher importance ranking can cache contents with higher popularity rankings. For example, if a router has $r(x) = 1$ on the delivery path, the expected ranking of content that can be stored on it is $1 \sim \frac{M}{N}$. More generally, the expected ranking of content that will be cached on the router with ranking $r(x)$ is $(r(x) - 1) \times \frac{M}{N} \sim r(x) \times \frac{M}{N}$. We use β to represent the cache probability for the content the ranking of which is in this range (In this case, the popularity ranking of the content is thought to be matched with the importance ranking of the node.). However, when the popularity ranking of the content is not in this range, the router will cache it with different probabilities as it is not matched with the importance ranking of the node. Let α represent the cache probability for contents with a higher ranking than $(r(x) - 1) \times \frac{M}{N}$. Let γ represent the cache probability for contents with a lower ranking than $r(x) \times \frac{M}{N}$. $P_x(c)$ therefore can be further described in Formula (1).

We can change α, β, γ to make routers cache data packets with different popularity rankings. In particular, when α and γ are set to zero, each node on the delivery path will store an equal amount of data packets with different rankings. For example, the node with the highest ranking will only cache data packets with the highest ranking $1 \sim \frac{M}{N}$. In our experiment, we will set β to 1 to guarantee that each node will cache the data packet whose popularity ranking is matched with its ranking definitely, and analyze how various values of α and γ impact the system performance.

$$P_{x,PRIRM}(c) = \begin{cases} \alpha & r(c) < \dfrac{(r(x)-1)\times M}{N} \\ \beta & \dfrac{(r(x)-1)\times M}{N} \le r(c) \le \dfrac{r(x)\times M}{N} \\ \gamma & r(c) > \dfrac{r(x)\times M}{N} \end{cases} \tag{1}$$

$$P_{x,LCE}(c) = 1 \qquad \forall r(c),\ \forall r(x) \tag{2}$$

$$P_{x,BC}(c) = \begin{cases} 1 & \forall r(c),\ r(x) = 1 \\ 0 & r(x) \neq 1 \end{cases} \tag{3}$$

$$P_{x,ProbCache}(c) = \frac{\sum_{i=1}^{N-(h_x-1)} C_i}{10 \times C_x} \times \frac{h_x}{N} \tag{4}$$

Formulae (2) and (3) describe the situation for LCE and BC respectively. We find that Formula (1) can also be generalized to LCE and BC. Specifically, LCE can be implemented by setting α, β, γ to 1 no matter what $r(x)$ and $r(c)$ are. BC is the case where $r(x)$ is equal to 1 and α, β, γ are set to 1. As a comparison, we also briefly describe the ProbCache [22] strategy here, which distributes the content along the path based on the path capacity and the distance from the content provider. Specifically, each router will cache data packets with a probability defined in Formula (4), where C_i is the cache size of the i-level node and h_x is the number of hops from the router x to the content provider. One key characteristic of this strategy is that it can push contents to routers closer to the consumer. However, later experiments show that our ranking-matched distribution scheme can result in a lower average hop count than this distribution strategy while providing the comparable server cache hit.

It is important to highlight that there are two main differences between our work and the recent literature. First, routers in our PRIRM strategy do not need to work collaboratively when implementing a ranking-matched cache decision. In contrast, [26] also proposes a scheme to place content replica on different nodes according to its popularity ranking. In this scheme, each router needs to spread the request vector from the access routers to gateways, which requires extra co-ordination overhead. Second, PRIRM strategy places popular contents on routers with high important rankings. However, [14, 25] will push popular contents to nodes closer to users without considering the node importance.

3.2.2. *Node importance ranking.* In graph theory, betweenness centrality which emerges in social network first [27] is a measure of centrality in a graph based on the shortest paths. It can be used to evaluate the importance of a node. In this sense, a node with higher betweenness centrality will have higher ranking. If $G = (V, E)$ is a non-directed graph with n nodes, then the betweenness centrality of node v can be defined in Formula (5).

$$Betw(v) = \sum_{s \neq v \neq t \in V} \frac{\theta_{st}(v)}{\theta_{st}} \tag{5}$$

where θ_{st} describes the total number of the shortest paths from node s to node t and $\theta_{st}(v)$ is the number of those paths that pass through node v. The betweenness centrality of each node can be computed when the topology is built up.

In order to get the ranking of a node on the delivery path, a BetWeenness Vector (BWV) field and a Cache Node Number (CNN) field should be added to the interest packet. BWV is used to record the betweenness centrality of all nodes which the interest packet passes through, and CNN keeps the total number of cache nodes on the delivery path. Each time when an interest packet passes by a node, CNN will be increased by one. In case of a cache hit, BWV and CNN field will be copied into the content packet. Hence, each router can compute its betweenness centrality ranking based on the BWV field when the corresponding data packet comes back.

3.2.3. *Content popularity ranking.* PRIRM scheme requires each router to know the content popularity ranking and the total distinct content population also. To achieve this goal, we propose a time window based method to estimate popularity ranking and content population. Specifically, each router will maintain a popularity table to record the request times for a content during a time window. Each table entry includes the content name, the associated request times and the predefined time window T, shown in Table 1.

Table 1. Content popularity table

Content name	Request times	Time window value
$content_1$	N_1	T
$content_2$	N_2	T
$\cdots$	$\cdots$	T
$content_m$	N_m	T

The popularity table will be updated each time when receiving an interest packet or the counting time reaches the predefined time window value. Specifically, when receiving an interest, the router will check if there is a record in the popularity table. If there exists a record, the associated request time will be increased by one. If not, the router will create an entry, set the request time to one and increase the total content population by one. As a result, when a content packet ct_x comes in, the router can lookup this table to get the request times. Assuming there are M records in this table and there are $L-1$ records which have larger request times than that of ct_x, we then can presume that the popularity ranking of ct_x is L. In such a situation, the most frequently requested content will have the highest popularity ranking. Meanwhile, the content category seen by this node can be estimated as M. After the counting time reaches the predefined time window value, we need to reset the request times for the associated content, let the total content population subtract one and begin to count for next time window. By taking this approach, we can estimate the popularity ranking of contents and the total distinct content population dynamically. Optionally, ndnSIM provides an application that requests contents following Zipf-Mandelbrot distribution. We can use the sequence number as the content name and it will represent the global popularity ranking of this content in the network. For example, a content with name (sequence number) "1" has the highest popularity ranking. As a result, a router can recognize the popularity ranking from the content name (sequence number) directly when receiving an interest or content packet. In latter experiments, we can find that our estimated popularity can result in a similar result as the global popularity.

Algorithm 1 Interest packet processing algorithm

```
Initialize (vector BWV = [0, 0, ..., 0], CNN = 0)
for V_k = i to j do
  if data in cache || entry in PIT then
    Get BWV, CNN
    if data in cache then
      copy BWV, CNN to data packet
      send(data)
    else
      create AggregatedFace(BWV, CNN)
      discard interest packet
    end if
  else
    get B(V_k)
    BWV_k = B(V_k)
    CNN = CNN + 1
    forward interest packet to next hop towards j
  end if
end for
```

Algorithm 2 Data packet processing algorithm

```
On receiving content ct_x:
for V_k = j to i do
  Get BWV, CNN
  Get B(V_k)
  Compute ranking of V_k
  Lookup ranking of ct_x
  Compute cache probability based on rank(V_k) and rank(ct_x)
  if face is an Aggregated Face then
    Get AggregatedFace(BWV, CNN)
    (BWV, CNN) = AggregatedFace(BWV, CNN)
    Send data packet from the face
  else
    Send data packet from the face
  end if
end for
```

3.3. **Packet processing algorithm.** In this section, we describe how each router on the forwarding path processes interest/data packets. We also give a specific example to make our PRIRM strategy easier to follow.

Algorithm 1 describes the actions taken by a node, denoted by V_k, when receiving an interest packet. There are three different cases:

i) This interest packet cannot be satisfied on V_k and there is no related PIT record. In this case, V_k will add the betweenness centrality of it to BWV field and CNN field will be increased by one. Then the packet will be forwarded to the next hop towards the content server;

ii) This interest packet can get a cache hit on V_k. In this case, CNN field and BWV field will be copied from it to the data packet, which will be forwarded out from the ingress port of this interest packet;

iii) This interest packet has a matching PIT entry on V_k, meaning that a similar request has been forwarded to V_k previously. In this case, V_k needs to store the incoming face of it, denoted by $Face_x$, in an existing PIT record rather than create a new entry. According to PRIRM strategy, the router should store BWV field and CNN field of the interest packet as well, and leverage them to replace corresponding fields of the response data packet when forwarding it out from $Face_x$. Our method is to set a flag field which is associated with the incoming face field in the PIT record. It can be used to judge whether the incoming face is aggregated or not. Specifically, if this is the first time the interest arrives, the flag field is set to a false value and the incoming face of this interest is not thought to be an aggregated face; otherwise, the flag field is set to a true value, and BWV and CNN field of this interest will be stored along with the incoming face in the PIT record. Finally, V_k can discard this interest packet.

Algorithm 2 describes the actions taken by a node, denoted by V_k, when receiving a data packet. V_k will first extract the BWV and CNN field from the content packet. Then it will look up the popularity ranking of the content in its popularity table and compute its betweenness centrality ranking based on the BWV field. This data packet will be stored with the probability defined in Formula (1). Afterwards, V_k needs to judge whether the incoming face in the PIT record is an aggregated face. According to whether this face is aggregated or not, there are two different cases:

i) This face is an aggregated face (i.e., the flag field has a true value). In this case, BWV and CNN field of the data packet will be replaced by the values associated with the aggregated face before it is forwarded out. This is important because we need to guarantee that the data packet can carry information of nodes that the aggregated packet passes through;
ii) This face is not an aggregated face. In this case, V_k does not need to change the data packet when forwarding it out from this face.

Figure 1 gives a specific example. Assuming that all caches are empty at the initial state, when v4 issues a request to retrieve content A (the popularity ranking of A is denoted by R_A), this request will finally arrive at server via v3 $\rightarrow$ v2 $\rightarrow$ v1. Then content A will be routed from the server to v4. Given that there are three nodes on the delivery path and the betweenness centrality ranking for these nodes is v2 $>$ v1 $>$ v3, content A will be stored on node: i) v2 if R_A is at the top 1/3; ii) v1 if R_A is within the range $1/3 \sim 2/3$; iii) v3 if R_A is at the last 1/3.

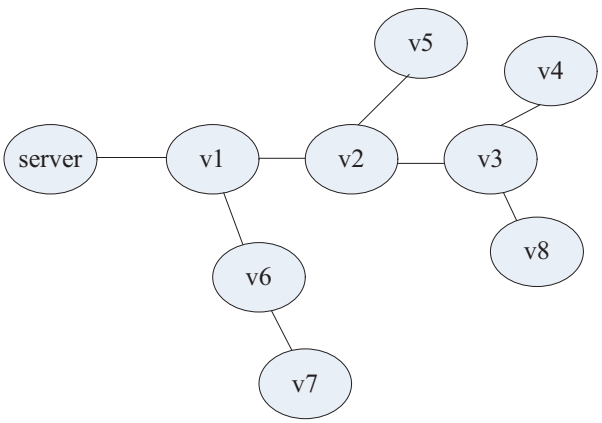


FIGURE 1. A specific example for PRIRM strategy

3.4. **Characteristics of PRIRM.** The core of PRIRM strategy is to distribute contents on the delivery path based on the popularity ranking of the content and the betweenness centrality ranking of the router. On the one hand, PRIRM scheme requires that each node should know the total number of caching nodes and betweenness centrality of each router on the delivery path. This information can be obtained by adding a BWV field and a CNN field in the interest packet. As the data packet carries the BWV field, each router can compute its ranking on the delivery path when the data packet comes back. On the other hand, PRIRM scheme requires that each node should know the popularity ranking of the content. Each router needs to maintain a popularity table in order to estimate the popularity ranking of each content and the total distinct content population.

In a static topology structure, the betweenness centrality of a router can be precomputed offline and stored on it. Since each router on the delivery path will make a caching decision without requiring information exchange with other routers, there is no communication overhead. Besides, the computation of cache probability is simple and will not put a severe burden on the caching process of a router. PRIRM strategy leverages the popularity ranking of content and the betweenness centrality ranking of the router to achieve efficient data distribution.

4. **Experiment and Result.** To demonstrate the advantage of our PRIRM strategy, we compare it with the following three benchmark caching strategies.

- **LCE**: According to this strategy, a copy of the requested content will be replicated at every router when the content is traversing on its way to the user [28, 29].
- **BC**: Content objects will be cached at the node with the highest betweenness centrality along the content delivery path [12].
- **ProbCache**: This strategy will cooperatively cache contents on a router considering the path capacity and the distance from the content provider [22].

4.1. **Simulation setting.** Our machine is Inter(R) Xeon(R) E5-2367 CPU@3.50GHz with 32G RAM. The operation system is Ubuntu 14.04 LTS 64 bit. We use ndnSIM [15], which is an open source NS-3 based simulator and faithfully implements the basic components of an NDN network in a modular way. By modifying the core packet processing, we realize our PRIRM strategy and these three benchmark strategies.

We use two metrics, i.e., average hop count H_t and server cache hit C_t over the simulation time period t to evaluate the performance of a caching strategy. H_t and C_t are defined in Formulae (6) and (7), where Q represents the total number of requests, h_r describes the hop count from the consumer to the first node where a cache hit occurs and w_r indicates whether a request is satisfied by the server. Specifically, if the request is satisfied by in-network caching, w_r equals 0; otherwise, it equals 1 (i.e., cache hit occurs on the server). We can see that if there is no in-network caching, $C_t = 1.0$. Basically, a caching strategy with a smaller H_t and C_t can reduce the traffic and improve user experience. It is worth mentioning that the decreasing of H_t is not proportional to C_t because the location where a request can be satisfied can be far from the server or close to the server. In these two situations, C_t is the same but the former will have a lower H_t value. Hence, the above two metrics are necessary to give an overall evaluation on the performance of a caching strategy.

$$H_t = \frac{\sum_{r=1}^{Q} h_r}{Q} \tag{6}$$

$$C_t = \frac{\sum_{r=1}^{Q} w_r}{Q} \tag{7}$$

Both regular topology and non-regular topologies will be used in our simulation: K-ary trees which have almost strict regular structure and scale-free Barabasi-Albert (BA) [30] graphs which are the most known generative model for power law distribution.

The total number of different content items is set to 1000 and the request rate follows the Poisson distribution with λ equal to 100. Content requests are generated following Zipf-Mandelbrot distribution [31]. The distribution parameters s and q are set to 0.7 to represent a normal network traffic [18]. Routers have a uniform cache store size = 10% of the total content population. The total simulation time is 300. Contents in a CS are replaced using a least recently used (LRU) algorithm [32]. We take advantage of the global routing controller and install corresponding FIBs on every node. As a result, an interest packet will be routed to the content provider directly rather than using the flooding strategy, which has been proven to be catastrophic regarding overhead. The simulation runs with the aforementioned parameters, unless otherwise specified.

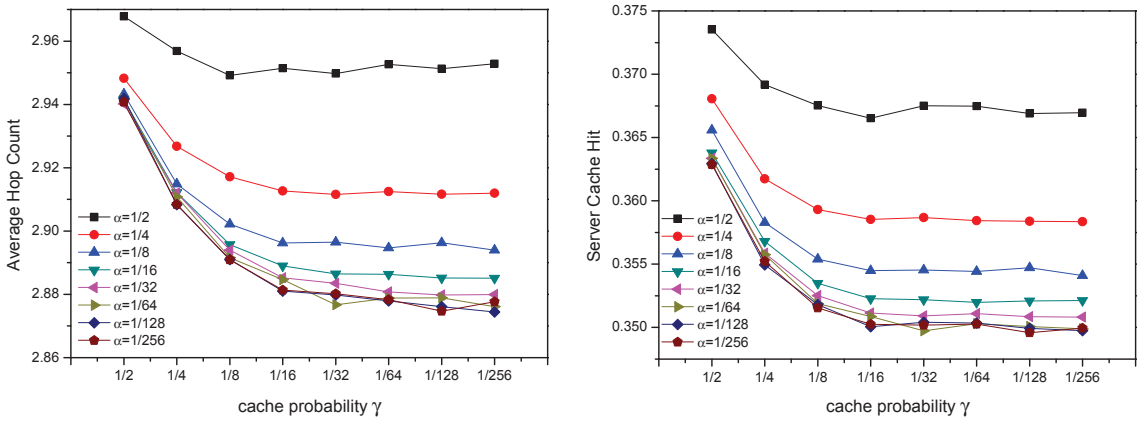
4.2. **Experiment on regular topologies.** We first evaluate the performance of our PRIRM strategy on a regular topology. A K-ary tree topology has two parameters, i.e., D and K. D describes the depth of the tree and K determines the child number of each node. The default experimental topology is a 6-level binary-tree with K equal to 2 and D equal to 6 (127 nodes in total). The root of the tree is the content server, which stores all the content. We configure requests to come from the last two levels like [22]. If the request reaches at the server, the server will respond it with a data packet.

4.2.1. *PRIRM parameters tuning.* As discussed in Section 3.2.1, α, β and γ determine the content category on a router. In order to make full use of cache capacity on a router, β is set to 1 to guarantee that each router will cache a data packet the popularity of which is matched with its ranking definitely. However, it will cache other content packets with probability α or γ. In this experiment, we set the time window to 10 seconds to make an estimation on content popularity and content population. We will analyze how α and γ impact system performance. Intuitively, when a router caches a data packet whose popularity ranking is not matched with its ranking, it will introduce redundancy between different nodes and in effect, reduce content diversity as there leaves no space for ranking-matched content packets. Hence, a large value of α or γ can very likely deteriorate the performance.

In Figure 2 we show experimentally measured values of average hop count and server cache hit against various values of α and γ, among which 1/2 represents a relatively high probability while 1/256 depicts a low probability. In these two sub-figures, the upper curve and the bottom curve have the maximum and minimum α value respectively and γ decreases gradually from the left to the right along the x-axis. Hence, α and γ show a decreasing trend from the top left corner to the right bottom. As it can be immediately noted from these two sub-figures, decreasing α and γ will improve average hop count and server cache hit for a value of α larger than 1/32. When α and γ are both smaller than 1/32, the average hop count and server cache hit show little difference for various settings. Considering that a nonzero value of α or γ will result in cache replacement[1] when the content store is full, we set both α and γ to zero in our strategy with the goal of both reducing the frequency of cache replacement and improving system performance.

[22, 33] show that caching the data packets within the first 10 seconds can effectively reduce redundant traffic based on today's technology. It can be a good starting point to set the time window for 10 seconds in our PRIRM strategy. Moveover, we measure the

[1]Frequent cache replacement can bring about extra processing overhead, which is not appropriate for high-speed ICN routers.



(a) Average hop count vs. α and γ (b) Server cache hit vs. α and γ

FIGURE 2. Cache performance for various values of α and γ

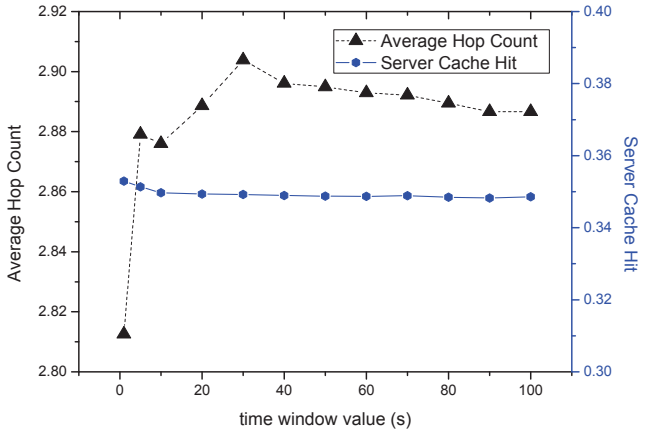
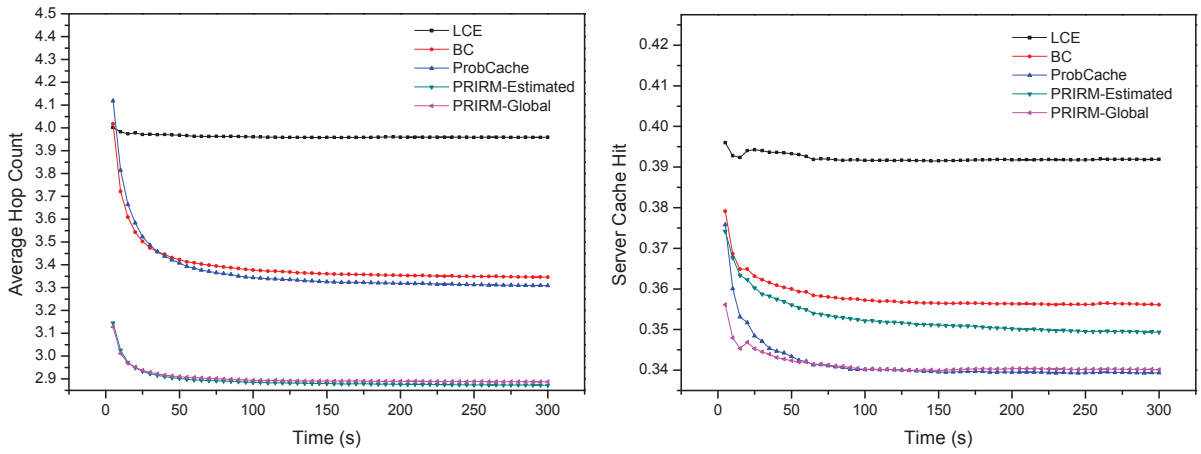


FIGURE 3. Cache performance when time window varies

corresponding average hop count and server cache hit when time window value changes. Results are shown in Figure 3. As it can be noted from the outer graph, server cache hit remains almost unchanged when time window value varies. The average hop count for a time window of 10 seconds is only 0.06 hops larger than the lowest value (the time window is set to 1 second in this case, but it results in the highest server cache hit). For this reason, it is reasonable to set the time window to 10 seconds. We will use it for executing all the following experiments.

4.2.2. *Instantaneous behavior.* Section 3.2.3 devises a method to estimate content popularity and content category by maintaining a popularity table. In fact, we can modify ndnSIM to get the global popularity from the content index directly and let each node know the total distinct content population in advance. Now we analyze the instaneous performance of our time window based method and global popularity. We call them as PRIRM-Estimated and PRIRM-Global respectively. Results are shown in Figures 4(a) and 4(b). After some time, simulation reaches the balance state for these algorithms. PRIRM-Estimated and PRIRM-Global perform much better than LCE, BC and Prob-Cache in terms of average hop count. Compared with PRIRM-Estimated, PRIRM-Global can reach the stable state faster. However, it is difficult to get this value directly in the real world. One insightful observation is that PRIRM-Estimated can result in a smaller



(a) Instaneous average hop count (b) Instaneous server cache hit

FIGURE 4. Instantaneous behavior of caching strategies

average hop count than PRIRM-Global. As PRIRM-Estimated considers the request distribution on the delivery path rather than the whole network, interest packets have a higher chance to get a cache hit by traveling smaller hop counts than that of PRIRM-Global. From the perspective of server cache hit, both ProbCache and PRIRM-Global show the best performance, but the performance gap between PRIRM-Estimated and them is merely 1%. It follows that our PRIRM-Estimated can result in a considerable performance as PRIRM-Global and it is enough to be used in our algorithm. In the following sections, we use PRIRM to refer to PRIRM-Estimated for simplicity.

4.2.3. *Hierarchical content distribution.* According to PRIRM scheme, each router will store a data packet whose popularity ranking is matched with its ranking in the CS. We first of all get a glimpse of how contents are distributed on the nodes at the steady state. As we adopt a K-ary tree topology with K equal to 2 and D equal to 6, there are six different rankings of nodes without considering the root node. We divide content population into six different ranking regions and then count the number of contents having the same ranking interval on the nodes with the same betweenness centrality ranking. Finally, we compute the percentage of contents in each ranking interval. Results are shown in Table 2.

There are two main observations from these results. First, our PRIRM strategy can result in a hierarchical content distribution. In fact, we find that: 1) nodes with the highest ranking has the largest percentage of contents with rankings 1∼166; 2) nodes with the second highest ranking has the largest percentage of contents with rankings 167∼332; and so forth. The top five content categories fit with this situation basically. This shows that contents are indeed distributed on nodes based on popularity rankings and node importance ranking in PRIRM strategy: important nodes store contents with higher popularity rankings and less important nodes store contents with lower popularity rankings. Therefore, our PRIRM caching scheme can result in a hierarchical cache. Second, contents do not show the same distribution for these three benchmark strategies. This is because they distribute contents on routers without considering the relationship between content popularity and node importance. Consequently, nodes having different rankings can cache data packets with different popularity somewhat. The popular and unpopular contents are mixed on different nodes in the network. Following experiments

TABLE 2. Content category on nodes with different rankings at steady state for PRIRM, BC, LCE and ProbCache strategy

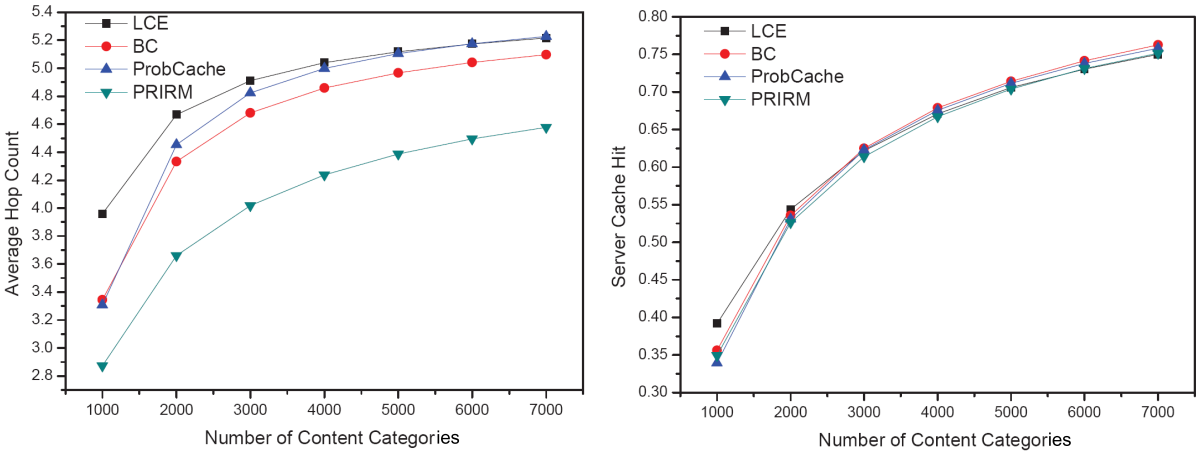
Caching Algorithm	Node Ranking	1∼166	167∼332	333∼498	499∼664	665∼830	831∼1000
PRIRM	1	**0.825**[1]	0.175	0	0	0	0
	2	0.517	**0.398**	0.085	0	0	0
	3	0.403	0.241	**0.209**	0.113	0.03	0.005
	4	0.376	0.212	0.141	**0.113**	0.094	0.065
	5	0.42	0.174	0.14	0.102	**0.09**	0.073
	6	0.625	0.142	0.088	0.063	0.046	0.035
BC	1	0.175	0.22	0.215	0.135	0.12	0.135
	2	0.235	0.245	0.193	0.128	0.11	0.09
	3	0.285	0.266	0.186	0.111	0.092	0.059
	4	0.419	0.272	0.158	0.081	0.049	0.022
	5	0.659	0.206	0.08	0.034	0.014	0.006
	6	0.82	0.126	0.037	0.012	0.003	0.002
LCE	1	0.265	0.17	0.2	0.13	0.14	0.095
	2	0.27	0.188	0.198	0.13	0.122	0.092
	3	0.289	0.195	0.181	0.121	0.122	0.091
	4	0.323	0.199	0.152	0.126	0.115	0.086
	5	0.373	0.198	0.147	0.109	0.094	0.079
	6	0.439	0.181	0.125	0.102	0.085	0.069
ProbCache	1	0.205	0.25	0.15	0.175	0.11	0.11
	2	0.22	0.223	0.16	0.14	0.13	0.128
	3	0.29	0.203	0.146	0.136	0.136	0.089
	4	0.327	0.203	0.164	0.125	0.093	0.087
	5	0.44	0.192	0.127	0.093	0.08	0.068
	6	0.594	0.135	0.089	0.075	0.058	0.049

[1] This value is larger than other values in the same column in PRIRM algorithm, meaning that nodes with ranking 1 have the largest percentage of contents with popularity rankings 1∼166.

will show that this kind of content distribution can yield a smaller average hop count, thus improving user perceived latency.

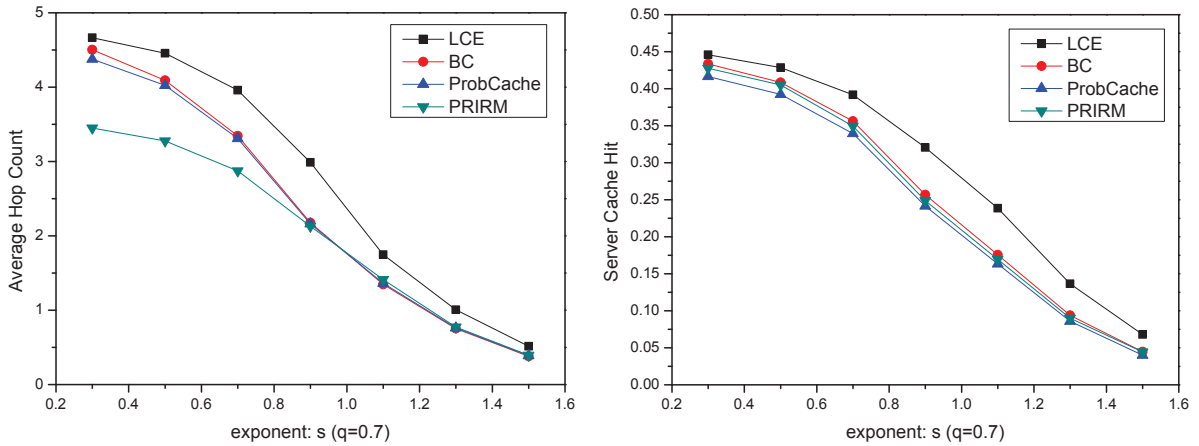
4.2.4. *Sensitivity analysis.* Content population and popularity distribution have a direct impact on the content distribution in PRIRM strategy. Now we study the sensitivity of PRIRM algorithm against these parameters. Results are shown in Figures 5 and 6.

Figures 5(a) and 5(b) show how content population affects the performance. Since the cache size of CS is fixed to 100, each router can only hold 100 data packets. A larger content population therefore results in a more frequent cache placement. Average hop count and server cache hit therefore both increase as the number of content population increases for these four caching schemes. Nevertheless, PRIRM outperforms these three benchmark strategies all the time in terms of average hop count. As the content population increases, contents are distributed on the nodes at a larger interval ranking. Each node still only caches data packets whose popularity rankings are matched with its ranking. Hence, cache replacement occurs less frequently than these three benchmark strategies. Interest packets thus will have a higher chance to get a cache hit at nearby nodes. In



(a) H_t against content population (b) C_t against content population

FIGURE 5. Cache performance when content population varies



(a) H_t against Zipf-Mandelbrot exponent s (b) C_t against Zipf-Mandelbrot exponent s

FIGURE 6. Cache performance when Zipf-Mandelbrot exponent s varies

general, average hop count in PRIRM is reduced by at least 0.5 hop compared with these benchmark strategies. In terms of server cache hit, these strategies show the considerable performance due to the limited path capacity. Part of requests cannot be satisfied by in-network caching.

Figures 6(a) and 6(b) present the average hop count and server cache hit against Zipf-Mandelbrot exponent s respectively. The most insightful observation is that PRIRM outperforms the other three policies in terms of average hop count when s is below 0.9, which is exactly the characteristic of web content popularity [18]. As LCE caches every packet, packets with different popularity are mixed and the most popular content in the CS has a high probability to be replaced by unpopular content. Hence, subsequent interest packets have to travel farther distances before hitting a copy of the requested content. BC strategy only caches data packets in the most important node, which can also lead to a frequent replacement as numerous interest packets pass by it. ProbCache strategy distributes the content based on the path capacity and the distance from the server without considering the content popularity. High popular contents have a high chance to be replaced by unpopular contents at the node closer to consumers. On the

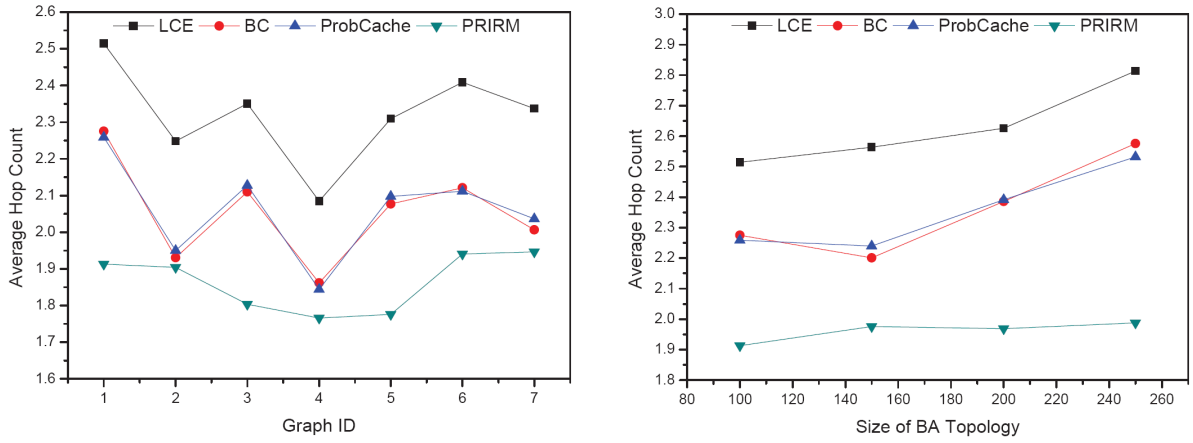
contrary, PRIRM distributes contents hierarchically based content popularity ranking and node importance ranking. Cache replacement occurs less frequently and popular contents can stay longer. When s is larger than 1.1, these four strategies get the similar results. We think this insightful observation is consistent with a high skewed content popularity distribution since requests are concentrated to retrieve some most popular contents. In this situation, the total cache size of the path from the users to the server can store a large percentage of the content population, most requests therefore can be easily satisfied by in-network caching. In terms of server cache hit, our PRIRM strategy provides a comparable performance to ProbCache strategy, which has the lowest value. In a word, in spite of resulting in a considerable server cache hit with ProbCache strategy, our PRIRM strategy has a lower average hop count value, especially when the Zipf-Mandelbrot exponent is smaller.

To summarize, the sensitivity analysis against different parameters verifies the effectiveness of PRIRM strategy on regular topologies in terms of average hop count.

4.3. **Experiment on scale-free topologies.** Most real networks are scale-free and more complicated than a K-ary tree topology. In fact, BA graphs can reflect better real network topologies than K-ary trees [12]. It is necessary to evaluate the performance of our PRIRM strategy on these scale-free topologies which can emulate the real networks. In this experiment, we randomly choose half of the network nodes as the consumers. The node with the middle betweenness centrality ranking is set as the content provider.

We first use BRITE [34] to generate several BA graphs with the same size to evaluate how different network topologies influence the performance. These graphs have the following settings: $N = 100$, mean valence $= 2$, growth type is incremental. Since each edge is attached to a vertex randomly with a probability directly proportional to its degree (also known as "preferential attachment"), each generation of a BA graph with the same settings results in a different topology. We will evaluate these four caching schemes over these different topologies like in [12]. Results are shown in Figure 7(a). We can see that our PRIRM strategy has a lower average hop count than LCE, BC and ProbCache in these randomly generated topologies.

Then, we increase the size of the topology and study how the size of the topology affects the performance. Figure 7(b) describes the experimental results. As it can be seen from the figure, our PRIRM caching scheme still provides the least average hop count as



(a) Caching performance in different BA graphs with 100-node

(b) Caching performance when the size of BA graph varies

FIGURE 7. Experiments on scale-free BA graphs

the size of the topology increases. These two experiments further demonstrate that our PRIRM caching strategy still has a significant advantage compared with LCE, BC and ProbCache on scale-free topologies.

4.4. **Discussion.** From the above experiments, we can find that our PRIRM strategy performs better than these three benchmark strategies in terms of average hop count, which can be used to indicate the number of links an interest packet traverses. A smaller average hop count means that a request can travel fewer links to get a cache hit. Therefore, the redundant network traffic (i.e., requests) can be reduced. Our PRIRM strategy can also reduce cache redundancy between different in-network nodes since contents are distributed on content popularity ranking and node importance ranking. This is exactly the key point why this strategy is efficient.

5. **Conclusion and Future Work.** Cache decision is an active and hot research area in ICN caching, which governs caching performance. Motivated by hierarchical caching in computer system and Web applications, we propose a novel ranking-matched caching strategy, i.e., PRIRM strategy. Our goal is to reduce cache redundancy and improve content diversity on the delivery path. The key characteristic of PRIRM strategy is to distribute contents based on content popularity ranking and node importance ranking. Since each node can make independent caching decision, there is no communication overhead. The sensitivity of our PRIRM strategy against different parameters (content number, popularity distribution, network topologies, etc.) is thoroughly studied. Experiments in ndnSIM simulator verify that PRIRM strategy has a lower average hop count than LCE, BC and ProbCache strategies on both regular and scale-free topologies.

In the future, content traces from the most common dataset (e.g., Rocketfuel dataset) will be used to feed a request generator and be used in our simulations. We will also implement experiments in real systems, such as Sea Computing [35], to verify the effectiveness of our PRIRM strategy further.

Acknowledgment. This work was supported by "The Next-Generation Broadband Wireless Mobile Communications Network" National Science and Technology of Major Projects (No. 2017ZX03001019). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

REFERENCES

[1] P. Stuckmann and R. Zimmermann, European research on future Internet design, *IEEE Wireless Communications*, vol.16, no.5, pp.14-22, 2009.

[2] H. Xu, Z. Chen, R. Chen and J. Cao, Live streaming with content centric networking, *The 3rd International Conference on Networking and Distributed Computing*, pp.1-5, 2012.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang and B. Zhang, Named data networking, *ACM SIGCOMM Computer Communication Review*, vol.44, no.3, pp.66-73, 2014.

[4] C. Severance, Van Jacobson: Content-centric networking, *IEEE Computer*, vol.46, no.1, pp.11-13, 2013.

[5] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros and G. C. Polyzos, A survey of information-centric networking research, *IEEE Communications Surveys and Tutorials*, vol.16, no.2, pp.1024-1049, 2014.

[6] A. V. Vasilakos, Z. Li, G. Simon and W. You, Information centric network: Research challenges and opportunities, *Journal of Network and Computer Applications*, vol.52, pp.1-10, 2015.

[7] G. Zhang, Y. Li and T. Lin, Caching in information centric networking: A survey, *Computer Networks*, vol.57, no.16, pp.3128-3141, 2013.

[8] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee and S. Yang, Advertising cached contents in the control plane: Necessity and feasibility, *Proc. of IEEE INFOCOM*, pp.286-291, 2012.

[9] L. Saino, I. Psaras and G. Pavlou, Hash-routing schemes for information centric networking, *ACM SIGCOMM Workshop on Information-Centric Networking*, pp.27-32, 2013.
[10] V. Sourlas, I. Psaras, L. Saino and G. Pavlou, Efficient hash-routing and domain clustering techniques for information-centric networks, *Computer Networks*, vol.103, pp.67-83, 2016.
[11] N. Laoutaris, S. Syntila and I. Stavrakakis, Meta algorithms for hierarchical web caches, *IEEE International Conference on Performance, Computing, and Communications*, pp.445-452, 2004.
[12] W. K. Chai, D. He, I. Psaras and G. Pavlou, Cache "less for more" in information-centric networks, *International IFIP TC 6 Conference on NETWORKING*, pp.27-40, 2012.
[13] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Y. Zhang and L. Dong, Popularity-driven coordinated caching in named data networking, *Proc. of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp.15-26, 2012.
[14] K. Suksomboon, S. Tarnoi, Y. Ji and M. Koibuchi, Popcache: Cache more or less based on content popularity for information-centric networking, *Local Computer Networks*, pp.236-243, 2013.
[15] A. Afanasyev, I. Moiseenko and L. Zhang, *ndnSIM: NDN Simulator for NS-3*, Technical Report NDN-0005, 2012.
[16] P. G. Rodriguez, C. Spanner and E. W. Biersack, Analysis of web caching architectures: Hierarchical and distributed caching, *IEEE/ACM Trans. Networking*, vol.9, no.4, pp.404-418, 2001.
[17] D. Perino and M. Varvello, A reality check for content centric networking, *ACM SIGCOMM Workshop on Information-Centric Networking*, pp.44-49, 2011.
[18] M. Yamamoto, A survey of caching networks in content oriented networks, *IEICE Trans. Communications*, vol.E99.B, no.5, pp.961-973, 2016.
[19] N. Laoutaris, H. Che and I. Stavrakakis, The LCD interconnection of LRU caches and its analysis, *Performance Evaluation*, vol.63, no.7, pp.609-634, 2006.
[20] V. Martina, M. Garetto and E. Leonardi, A unified approach to the performance analysis of caching systems, *Proc. of IEEE INFOCOM*, pp.2040-2048, 2014.
[21] S. Arianfar and P. Nikander, On content-centric router design and implications, *Re-Architecting the Internet Workshop*, pp.1-6, 2010.
[22] I. Psaras, K. C. Wei and G. Pavlou, Probabilistic in-network caching for information-centric networks, *Edition of the ICN Workshop on Information-Centric Networking*, pp.55-60, 2012.
[23] D. Rossi and G. Rossini, *Caching Performance of Content Centric Networks Under Multi-path Routing (and More)*, Relatório Técnico, Telecom ParisTech, 2011.
[24] C. Bernardini, T. Silverston and O. Festor, MPC: Popularity-based caching strategy for content centric networks, *IEEE International Conference on Communications*, pp.3619-3623, 2013.
[25] N. Abani, G. Farhadi, A. Ito and M. Gerla, Popularity-based partial caching for information centric networks, *Mediterranean Ad Hoc NETWORKING Workshop*, pp.1-8, 2016.
[26] W. Li, Y. Li, W. Wang, Y. Xin and T. Lin, A popularity-driven caching scheme with dynamic multipath routing in CCN, *Computers and Communication*, pp.633-638, 2016.
[27] L. R. Izquierdo and R. A. Hanneman, *Introduction to the Formal Analysis of Social Networks Using Mathematica*, University of California, Riverside, 2006.
[28] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. M. Maggs, K. C. Ng, V. Sekar and S. Shenker, Less pain, most of the gain: Incrementally deployable ICN, *ACM Special Interest Group on Data Communication*, vol.43, no.4, pp.147-158, 2013.
[29] E. J. Rosensweig and J. Kurose, Breadcrumbs: Efficient, best-effort content location in cache networks, *INFOCOM 2009*, pp.2631-2635, 2009.
[30] A. Barabasi and R. Albert, Emergence of scaling in random networks, *Science*, vol.286, no.5439, pp.509-512, 1999.
[31] Z. K. Silagadze, Citations and the Zipf-Mandelbrot's law, *Complex Systems*, vol.11, no.6, 1999.
[32] D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho and C. S. Kim, On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies, *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp.134-143, 1999.
[33] A. Anand, C. Muthukrishnan, A. Akella and R. Ramjee, Redundancy in network traffic: Findings and implications, *Measurement and Modeling of Computer Systems*, vol.37, no.1, pp.37-48, 2009.
[34] A. Medina, A. Lakhina, I. Matta and J. Byers, BRITE: An approach to universal topology generation, *Proc. of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp.346-353, 2001.
[35] J. Tian, Sea-cloud coordinative and look ahead (Preface), *Journal of Network New Media*, vol.3, no.1, p.1, 2014.