On Performance of Cache Policy in Information-Centric Networking

Sen Wang, Jun Bi, Jianping Wu
Network Research Center, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology (TNList)
Beijing, China
wangsen@netarchlab.tsinghua.edu.cn; {junbi, jianping}@cernet.edu.cn

***Abstract*—Information-Centric Networking is (ICN) [1] gaining increasingly concerns, as an important direction of the future Internet architecture research. To study the impacts of various cache policies on overall performance of ICN network, we formulate the in-network caching problem of ICN into Mixed-Integer Linear Programming problem. Furthermore, we infer that frequency-based cache policies like LFU are supposed to perform well by analyzing the properties of optimal cache assignment, which is corroborated by our simulation results. We attempt to explore the impact of the distance between cache and the original content on cache policy performance by posing a novel cache policy named LB (Least Benefit), which takes into account the distance factor besides frequency. Through extensive simulations under various scenarios and configurations, we find that the performance gain brought by LB is limited in comparison to that of LFU, which implies that more sophisticated cache policies involving the distance factor should be considered to improve LFU. Our simulation results also show that, under a reasonable setting of cache size and request pattern, the average hops to get content can be reduced significantly by nearly 50% in comparison to that of the scenario without in-network caching.**

Keywords-component; Information-Centric Networking; Caching; Caching Policy

I. INTRODUCTION

As a future Internet architecture proposal, ICN intends to motivate the architectural transition from today's host-centric Internet architecture to information-centric in order to disseminate efficiently and scalably the enormous informations generated by a variety of applications. In this research direction, many approaches have been proposed such as PSIRP [12], NetInf [13, 14], PURSUIT [10], CCN [15], DONA [16] and NDN [11]. In-network caching, which means that every router is enabled with the capability of caching content passing by, is considered as one of the most important properties of ICN, and could be leveraged to improve network performance significantly in terms of improving network utilization and reducing propagation delay. In this paper, we strive to study the impact of various cache policies on overall performance of ICN network.

It is well known that the property that content is requested unevenly by the Internet users results in that relatively more popular content objects are transmitted repeatedly, which affects the network utilization negatively [20]. In contrast to other caching approaches for redundancy elimination (e.g. Web caching, Packet-level Redundant Traffic Elimination), the caching issue of ICN discussed in this paper is based on explicit identifiers for content objects, en-route caching paradigm and the pub/sub communication paradigm. With the aim of being independent of any specific approaches aforementioned, in this paper, we extract the ICN structuring architectural properties and assumptions as the following. We refer to the content identifier as CID (content ID) which names a content object uniquely (e.g. a video or a named single packet). The content transmission mechanism adopts the Pub/Sub communication paradigm which is embedded in most of the approaches aforementioned. A request is sent to get a specific content object with the CID specified in its packet. The request is routed to the original content without awareness of any cached copy in routers by some routing mechanism (e.g. OSPF). We make this assumption because of considering that being aware of current caching state of the whole network would impose the routing system significant burden in terms of maintaining extra states. Along the path of a request, any router caching a copy of the desired content object could respond to this request with its copy. We assume the content object will go back to the requester along the same path which means an accurate symmetric routing. While forwarding a content object, an intermediate router can decide whether to cache the content object and evict some cached content objects to make space for this one according to its own cache policy. The goal of this paper is to study the performances of a variety of cache policies, e.g. LRU, LFU etc., concerning the bandwidth consumption and propagation delay.

So far, cache policy for ICN is barely explored. In [3], preliminary evaluation on network performance improvement by random autonomous caching is exhibited with very simple topology and scenario. In this paper, we consider the network performance benefit brought by in-networking caching and study the impact of employing different caching policies on the overall network performance of ICN. Our contributions are threefold:

1) Using average hops to get content as the metric of overall network performance, we formulate the optimal cache assignment into Mixed-Integer Linear Programming in which the optimal cache assignment can be calculated by general MILP solver.

2) The properties of the optimal cache assignment are explored by analysis. Concerning our hop-based metric, we find frequency-based cache policies like LFU are supposed to perform well, which is corroborated by extensive simulations.

978-1-4673-1544-9/12/$31.00 ©2012 IEEE

3) We explore the impact of the distance between cache and the original content on cache policy performance. We design a cache policy named LB (Least Benefit), which takes into account the distance factor. We find by extensive simulations under various scenarios and configurations that the performance gain brought by LB is limited in comparison to LFU, which implies that more sophisticated cache policies involving the distance factor should be considered.

The rest of the paper is organized as follows. Section 2 presents the Mixed-Integer Programming Formulation of the in-network caching problem of ICN. In Section 3, the properties of the optimal cache assignment are studied and a novel cache policy named LB is posed by taking the distance factor into account. In Section 4, we evaluate our analysis findings and the performance of various cache policies with extensive simulations based on NS3. We conclude in Section 6.

II. PROBLEM FORMULATION

For ICN network, one of the main concerns in our opinion is how to cache the content objects so that the network resource consumed or/and the propagation delay is/are minimized. Considering the stationary caching state of all the caches in ICN, the in-network caching problem can be described as the following. Given an ICN network, the request rate from each router to each content object, the cache capacity of each router, a set of initial content objects, an assignment of resident routers of these content objects and a fixed routing path from each router to each content, the goal is to find a feasible assignment of caching copies to routers in order to minimize the overall resource consumption of the network which is measured by average hop number to get desired content. Now we start to formulate this problem into a MILP problem as follows.

Graph construction:

The ICN network is represented as an undirected graph $G = (V, E)$. V is the set of nodes in the network. E is the set of edges in the network. $b: V \rightarrow R^+$ is a function. $b(v)$ denotes the storage capacity of the node v. C is the set of initial contents. A content object is denoted by a two-tuples (l, m), where l is the resident node of the content and m is the size. For any content object $c \in C$, function $l(c)$ and $m(c)$ return the resident node and the size respectively.

Traffic demand:

$q_{v,c}$ is the ratio between the number of requests initiated from node v for the content object c and the number of all the requests. Obviously, we have $\sum_{c\in C}\sum_{v\in V} q_{v,c} = 1$.

Routing path:

We assume that there is only one routing path between any pair of nodes. $p_{u,v}$ denotes the single path between node u and node v and the function $u(p_{u,v})$ gives the length of this path. The k-th node in the path from node u is denoted by the return value of the function $g(p_{u,v}, k)$ for any $0 \le k \le u(p_{u,v})$.

An assignment of in-network caching:

We use a series of binary variables $\delta_{c,v} \in \{0, 1\}$ to describe the caching state. If node v has cached the content object c in its storage, the $\delta_{c,v}$ has the value 1, otherwise 0.

For a request, the network resource consumed is estimated as the product of the number of the hops traversed to get the first copy of the desired content object and the size of corresponding content object. The overall objective function is as follows.

$$\min \sum_{v\in V}\sum_{c\in C} q_{v,c}\ m(c) \min_{0\le i<u(p_{v,l(c)}),\ \delta_{c,g(p_{v,l(c)},i)}!=0} \left(\delta_{c,g(p_{v,l(c)},i)}\ i,\ u(p_{v,l(c)})\right) \quad (1)$$

$\min_{0\le i<u(p_{u,v}),\ \delta_{c,g(p_{v,l(c)},i)}!=0} \left(\delta_{c,g(p_{v,l(c)},i)} i, u(p_{v,l(c)})\right)$ is the minimal number of hops from v to any copy of content object c along the routing path $P_{v,l(c)}$, including the original content object resided in node $l(c)$. The objective function (1) is subject to the storage capacity constraint of any node v:

$$\sum_{c\in C} \delta_{c,v} m(c) \le b(v), \forall v \in V \quad (2)$$

Note that the *min* function is not a linear function, so we involve a series of additional continuous variables $\mu_{v,c}$ and binary variables $\sigma_{v,c,i}$ for every node v and every content object c to linearize the objective function. The binary variables $\sigma_{v,c,i} \in \{0,1\}$ indicate whether the $\delta_{c,g(p_{v,l(c)},i)} * i$ is the minimal hop number for any i subject to $0 \le i < u(p_{v,l(c)})$. The $\sigma_{v,c,u(p_{v,l(c)})}$ represents the original content object. These variables are subject to the following constraint:

$$\sum_{i=0}^{i\le u(p_{v,l(c)})} \sigma_{v,c,i} = 1, \forall v \in V, \forall c \in C \quad (3)$$

Besides, we have the following constraint for the variables $\mu_{v,c}$:

$$\delta_{c,g(p_{v,l(c)},i)} \ge \sigma_{v,c,i}, \forall v \in V, \forall c \in C, 0 \le i < u(p_{v,l(c)}) \quad (4)$$

$$\mu_{v,c} \ge \delta_{c,g(p_{v,l(c)},i)}\ i - (\max_{v\in V, c\in C} u(p_{v,l(c)}))(1 - \sigma_{v,c,i}), \quad \forall v \in V, \forall c \in C, 0 \le i < u(p_{v,l(c)}) \quad (5)$$

$$\mu_{v,c} \ge u(p_{v,l(c)}) - (\max_{v\in V, c\in C} u(p_{v,l(c)}))\left(1 - \sigma_{v,c,u(p_{v,l(c)})}\right), \quad \forall v \in V, \forall c \in C \quad (6)$$

Inequality (4) indicates that when $\sigma_{v,c,i}$ has the value 1, the $\delta_{c,g(p_{v,l(c)},i)}$ can not have the value 0, which ensure that when a node v is indicated to be the minimal, it must have the copy. Inequality (5) associated with (6) assures that $\mu_{v,c}$ is at least larger than one of $\delta_{c,g(p_{v,l(c)},i)} * i$ for any i subject to $0 \le i < u(p_{v,l(c)})$ or the path length $u(p_{v,l(c)})$ and the minimization objective function can naturally ensure $\mu_{v,c}$ take on the minimal value.

Finally, we have the ultimate objective function:

$$\min \sum_{v\in V}\sum_{c\in C} q_{v,c}\ m(c)\ \mu_{v,c} \quad (7)$$

It is subjecst to the constraint (2)(3)(4)(5)(6).

III. CACHE POLICY

A. Analysis of optimal cache assignment

To explore the solution space of cache assignment for the optimization objective function (7), we first study the case in which there is only one requester in the network. In Figure 1, node *a* is the only requester. According to the hypothesis of our model that there is only one path for every pair of nodes in the network, the paths from node *a* to any other node form a tree rooted at node *a* as Figure 1 shows. For analytical brevity, we further assume all the content objects have the same size *cs* and the storage capacity of any node *v* is integral times of *cs*, namely $n_v * cs$. Let *d* denote the maximum hops from node *a* to any other node. We partition the set of all the nodes into *d* subset according to the node's distance to node *a*, namely $S_i, 0 \le i \le d$. S_i represents the set of nodes whose distance away from node *a* is *i*.

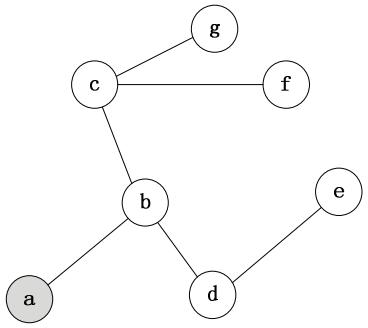

Figure 1. Spanning Tree originating from a requester in ICN.

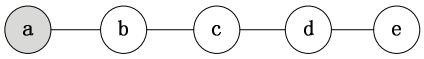

Figure 2. Linear topology

We first prove the following theorem.

Theorem 1: If an assignment *A* is among the optimal cache assignments, it must have the property that any content object cached in node *n*, its request rate is not smaller than that of any content object cached in nodes which reside in the subtree rooted at node *n*.

Proof: For contradiction, suppose such an optimal cache assignment exists so that there is at least one pair of content objects does not comply with that the request rate of one content object cached by root node of some subtree is smaller than that of the other content object cached by the nodes within the subtree.

In this assignment denoted by *A*, suppose one of these pairs of content objects is *d* and d' and they reside in the node $n(d)$ and $n(d')$ respectively. Their hop counts from node *a* are $L(d')$ and $L(d)$ respectively. For $n(d')$ is one of nodes within the subtree rooted at $n(d)$, $L(d')$ is larger than $L(d)$. According to the hypothesis, the request rate of content object d', denoted by $q_{a,d'}$, is larger than that of content *d*, denoted by $q_{a,d}$. For the content object d' is cached in node $n(d')$ within the assignment, the resident node of the original content object d' is in the subtree rooted at $n(d')$ according to the rational caching assignment definition. Then we can exchange the caching node of the pair of content objects and end up with a new caching assignment. The new assignment will get additional profit gain $(q_{a,d'} - q_{a,d})L(d')$ and profit lose smaller than $(q_{a,d'} - q_{a,d})L(d)$ (smaller in the case that the hop count of the resident node of the original content object d' from node *a* is larger than $L(d')$). Therefore the final profit gain will be larger than $(q_{a,d'} - q_{a,d})(L(d') - L(d))$, and the new assignment is found a more optimal one which is contradicted with that the former assignment *A* is an optimal one. The Theorem 1 is proved. ■

Let *Sub(n)* denotes the set of nodes within the subtree rooted at *n*. To generate a caching assignment complying with Theorem 1, we conceive a greedy algorithm as follows:

Algorithm 1: Greedy Caching

```
1:D = ∅;
2:for(i =0; i <= d; i++)
3:{
4:    E = S_i;
5:    Repeat
6:    select node n in E;
7:     for (k = 0; k <= size(n); k++)
8:     {
9:          find content c, so that l(c) ∈ Sub(n), c ∉ D, and
            q_{v,c} ≥ q_{v,c'}, for ∀c' ∈ C, c' ∉ D, l(c') ∈ Sub(n);
10:         caching c in the cache of n;
11:         D = D ∩ {c};
12:    }
13:   E = E − {n};
14:   Until E ≠ ∅
15:}
```

The Algorithm 1 greedily caches content objects with maximum request rate from the root node of the spanning tree to the leaf nodes. Obviously Algorithm 1 assures that for any content object cached in node *n*, the request rate is not smaller than that of any content object cached in nodes which reside in the subtree rooted at node *n*. However, we cannot prove that this greedy algorithm can generate optimal caching assignment. Take the special case shown in Figure 2 as an example. There is four content object, namely *{c1, c2, c3, c4}*, with the same size *cs* and request rate *{0.28, 0.25, 0.24, 0.23}* respectively. The resident node of original content object *c1* is *c* and the resident nodes of other content objects are the same node *e*. Node *b*, *c*, and *d* have a cache with storage capacity of one content object and other nodes have no cache. Obviously the optimal caching assignment is *{<c2, b>, <c3, c>, <c4, d>}*, whereas the assignment Algorithm 1 generates is *{<c1, b>,*

<c2, c>, <c3, d>}. The Algorithm 1 does not take into account the distance from the requesting node to the resident node of the original content object. We observe that whether a content object with higher request rate than that of any content object within a subtree should be cached in the root node of the subtree depends on whether the profit brought by caching it can offset the overall profit loss generated by replacing a series of content objects away further along some path from the root node to some leaf node to make caching space for it.

B. Cache policy considering distance factor

Now we discuss this problem from the perspective of specific cache policy. It is easy to find that the Perfect-LFU (relative to In-cache LFU, which records frequency only for items in the cache, whereas Perfect-LFU records frequency for all items it has ever seen even if those items have been swapped out) can result in the same caching assignment as the Algorithm 1 generates, considering static request rate. Although we have proved it is not definitely the optimal assignment, we expect that, in practice, the Perfect-LFU is a relatively good sub-optimal cache policy intuitively from the Theorem 1 proved above, given a relatively continuous request rate distribution and a relatively random resident node distribution of all the content objects. For the more general case of multiple requesters, diverse content size and diverse cache capacity, we expect the same suspect as the case with single requester.

Remaining the good property of Perfect-LFU consistent with Theorem 1, we expect to improve the Perfect-LFU by taking into account the distance from the hitting cache to the next node which has cached the desired content objects or the resident node of the original content object. Intuitively, we give each hit a weight which is equal to the hop gain by caching the content locally. We refer to the new cache policy as Perfect-LB (Least Benefit). We refer to the In-Cache version of Perfect-LB as InCache-LB, which does not need so much storage capacity for maintaining extra information, e.g. frequency and hop count, as Perfect-LB does.

To implement LB, each node needs to know the hop gain from caching some content *v* locally, for which we introduce an additional field into the ICN protocol, namely *HopGain*. The *HopGain* is in the content packet and indicates the current hop count it has traversed from the resident node of this copy of the content.

With this additional field, every intermediate node could know the hop reduction for any content object it has encountered if caching it locally. They will record the hop reduction (also referred to as distance) for every content object cached. When a hit happens, the current overall benefit of the hit content object would be increased by the current hop reduction. When a replacement is needed, the content object with least benefit will be evicted out.

Note that the hop reduction of a content object would change when the upstream node evict or a nearer upstream node cache the same content object. To perceive the hop reduction change, we make the additional change to ICN protocol. When a request is satisfied by a cached content object, it will go further upstream to the nearest next node which has cached the same content object with a probability of *p*. The further-going request will bring back no content object but just measure the newest hop reduction. Actually, in contrast to LFU, LB is a cooperative cache replacement policy in the sense that local cache decision will affect other caches significantly.

This Perfect-LB may impose enormous states maintained on routers as the Perfect-LFU does due to recording benefit for every content object encountered. We also implement a more practical version for Perfect-LB, which is referred as InCache-LB. InCache-LB only records the benefits for content objects in cache currently.

IV. EVALUATION

In this section, we present our simulation results of various caching policies applied in ICN. The primary evaluation metric is the average hop number of all the requests to get corresponding content objects. We implement a content-based transport protocol based on the Pub/Sub communication model of ICN and a series of caching policies on the open source network simulation software NS3. We consider both simple topology and practical ISP topology settings.

In the following simulations, several cache policies are involved besides the LFU and LB. In [2], various cache policies are evaluated in the case of a single Web cache with real HTTP traffic, the reader can refer to it with the details of cache policies. In particular, the difference between the Perfect-LFU and InCache-LFU is that Perfect-LFU would record the current frequency for every object it met even the object is evicted out. In contrast, the record would be deleted when the object is evicted out for InCache-LFU. Therefore, the Perferct-LFU is not feasible for its extremely large consumption of memory, but it is meaningful in case of analysis. The situation is the same for Perfect-LB and InCache-LB proposed in this paper.

A. Traffic Model for Evaluation

We argue that designed around content, the ICN features should be studied around contend-based request model. We expect the ICN traffic model has the same features as those of Web traffic model which prior studies have exhibited. In [4], Breslau et al. shows that the request rate of Web object follows the Zipf-like distribution, which means the request probability of the *i*th most popular object is proportional to $\frac{1}{i^{\alpha}}$, and independent reference model is sufficient for qualitative analysis. In this paper, the Zipf–Mandelbrot (also known as the Pareto-Zipf) distribution is used as the traffic model for simulation. The request rate distribution is represented as follows:

$$P(i) = \frac{\Omega}{(i+q)^{\alpha}} \quad (8)$$

The two parameters α and q in the equation above are the shape parameter and shift parameter respectively and Ω is the normalizing constant.

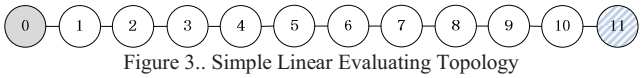

Figure 3.. Simple Linear Evaluating Topology

B. Evaluation with Simple Linear Topology

In this subsection, the results of a series of experiments under a simple linear topology are presented. Within the linear topology, 11 nodes form a line from node 0 to node 11 as Figure 3 shows.

Scenario of single content source:

In this scenario, a set of simulations is conducted with a single content publisher and a single requester, namely node 11 and node 0. We fix the size of all the content objects to 1 KB. We put 10,000 content objects at node 11 and the requester at node 0 which requests content objects with the popularity distribution indicated by equation (8). Each router has a cache with storage capacity of 500 content objects, except for node 11, where the overall cache size is up to 55% of the aggregate size of all content objects. The parameters of traffic pattern, namely α and q, are fixed to 0.7 which are reasonable for real traffic of Web request. The arrival of request follows the Poisson Process, which means that the time interval of two adjacent requests is exponentially distributed. Unless some changes of these parameters are specifically stated, all the following experiments are conducted under the same setting above.

First, we study the proportions of different hops to get desired content for request. In order to get reliable results, we make our simulator run 50,000 requests to warm up before we start the statistic of another 100,000 requests.

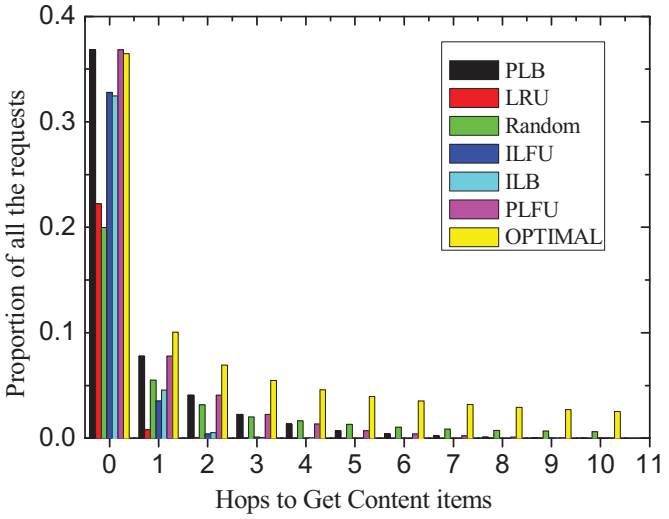

Figure 4. Request Proportions vs Hops

Figure 4 shows the hop distribution where requests find the desired content for different caching policies. It can be observed that the proportion of content objects decreases greatly while the hop number increasing. Within this simple scenario, the optimal caching assignment is trivial to be calculated, where the higher request rate a content object has, the closer it is placed to the only requester. The simulation result in Figure 4 shows that the Perfect-LB and Perfect-LFU are closer to the optimal than others, but they are still far from the amount of the optimal when the hop number exceeds 3 (only 41.2% of the optimal), while the proportions descent to almost zero for other non-perfect cache policies. This can be explained by the Filter Effect, which has been studied in [5]. After a Zipf-like request traffic goes through a cache, the resulting request pattern will be flatten in log-log coordination for a certain range of its most popular content objects, which make the following cache can hardly achieves high hit rate. This effect impacts the following simulation results dramatically.

We studied the effects of cache size and parameters of request pattern, namely q and α, on average hop number. The results are exhibited in Figure 5. Figure 5 (a) shows the effect of cache size, which varies from 27% of the aggregate size of all the content objects to 110%, for the aggregate cache size of all the nodes. Average hop number descent when cache size ascents, but the slope descents accordingly, which means simply increasing cache size cannot decrease average hop number efficiently. The parameter q of request pattern have little impact on average hops according to the plots in Figure 5 (b), while the parameter α does have a significant effect as Figure 5 (c) shows. The average hop number drops from 8.7 to 4.7 for InCache-LFU. For all the simulation results, the LB and LFU have the same performance on both metric due to the settings of simple topology and single requester.

Scenario of multiple requesters and content sources:

The scenario studied in this subsection differs from the previous one in that all the nodes can generate requests with the same request rate (we study the case of diverse request rate among nodes in the following subsection.), and the content objects are distributed randomly to the nodes instead of residing in a single node. Under this scenario, a series of simulation results is shown in Figure 6, which exhibits the different effects of parameter α, cache size, and topology scale. Almost the same impacts of both parameter α and cache size are observed in Figure 6 (a) and Figure 6 (b) as the prior subsection shows. With α and cache size to be fixed to 0.7 and 55% respectively, the InCache-LB gains 44.2% reduction in average hops compared to that in the case of no cache and slightly higher than InCache-LFU by 2.9%. We studied the impact of topology scale by increasing the node number from 12 to 24. The simulation results shown in Figure 6 (c) suggest that, as the topology scale becoming larger, the reduction percentage of average hops arises from 44.2% and 41.3% to 51.2% and 46.5 % for the InCache-LB and InCache-LFU respectively.

C. Evaluation with ISP Topology

A series of simulations using practical ISP topology is conducted to evaluate cache policies. First we use the PoP topology of ISP with AS No. 1221 provided by [9] for studying the effect of parameter α and cache size on average hop number. The simulation results are shown in Figure7 (a) and Figure 7 (b). Figure 7 (a) show that the parameter αdoes have a significant impact on average hops. While α increase from 0.5 to 0.9, the average hop number drops from 2.26 to 1.23, and the hop reduction ratio rises from 25.4% to 59.2% for LB. Figure 7 (b) shows the effect of cache size. The aggregate cache size of all the nodes varies from 55% of the total size of all the content objects to 220%. The average hop number descents when the cache size ascents, but the slope descents accordingly.

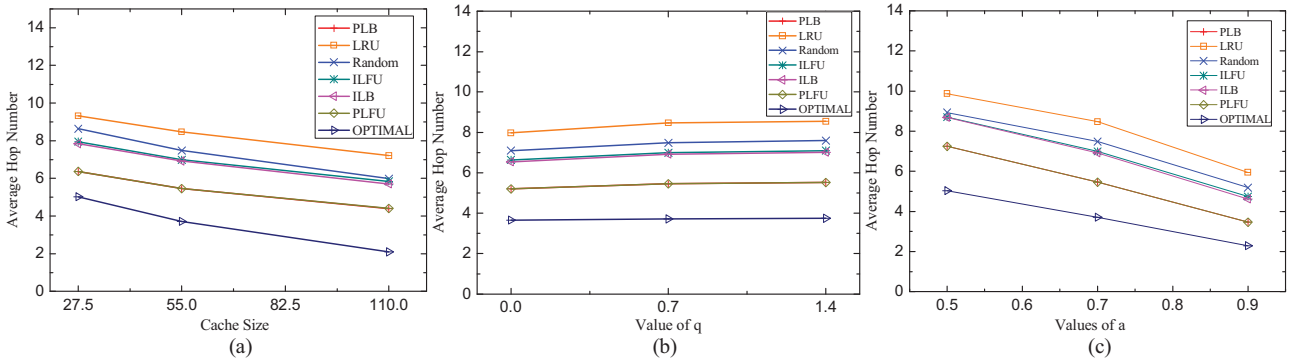

Figure 5. Impact of cache size and parameters of request pattern for simple topology with single requester

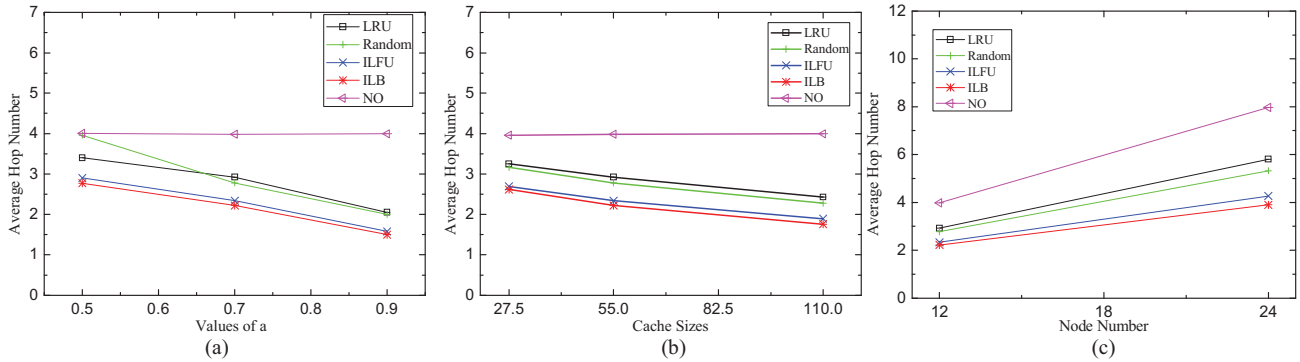

Figure 6. Simulation results for simple topology with multiple requesters

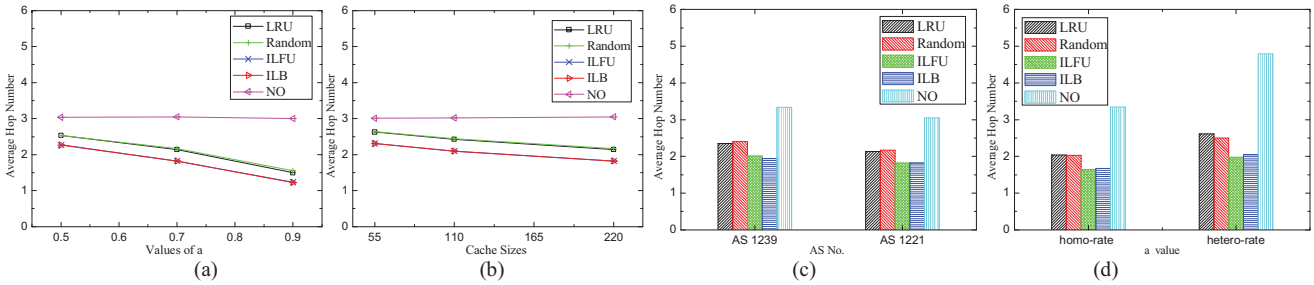

Figure 7. Simulation results for ISP topology

With α and cache size fixed to 0.7 and 110% respectively, the LB gains 30.5% reduction in average hop number compared to that in the case of no cache and has almost no obvious difference with LFU. It seems that LB does not perform obviously better than LFU, and we cannot improve performance by simply making use of the distance factor.

We explore the effect of different ISP topologies on the performance of various cache policies by simulating under the PoP topology of another ISP with AS No. 1239, which has 78 nodes and 84 edges, larger that AS 1221 consisting of 44 nodes and 44 edges. The Figure 7 (c) shows that the average hops of the five cache policies for the Pop topologies of the two ISPs. In this figure, it can be observed that different topologies result in almost no difference in average hop reductions, which are 40.3% and 41.7% for AS 1221 and AS 1239 respectively with the cache policy of LB.

We also studied the effect of heterogeneous request rate among nodes. In all the former simulation, each node request content objects with the same mean value of request intervals, namely one second. In contrast, the request rates of nodes range from 10 per second to 1 in this simulation. The simulation is conducted under the PoP topology of AS 1239. Figure 7 (d) shows the corresponding results, which suggests that, in the setting of heterogeneous request rates, more average hop reduction can be achieved, arising from 49.9% to 57.3%.

V. RELATED WORK

While ICN is gained increasingly concern recently, some research efforts are made to address its specific mechanisms including caching. In [19], Ghodsi et al. consider the performance gain of in-network caching as the most emergent issue to address for supporting the necessity of transition to ICN. In [3], Arianfar et al. focus on the packet-level caching in ICN and tests the caching replacement policy of random autonomous caching with flow-based testing traffic model under simple network topology. On contrary, this paper focuses on the impact of cache policies on the overall network

performance. In [17], Muscariello et al. proposes an analytical characterization of statistical bandwidth and storage sharing as well as a closed-form expression for average content delivery time. This work is based on analysis with assumptions of tree-like topology and single fixed cache policy LRU, which differs with our assumption of internet-like topology and a wide range of cache policies.

The property that contents are requested unevenly by the Internet users results in that relatively more popular contents are transmitted repeatedly, which impacts the network utilization negatively [20], To solve this problem, caching is involved and extensively explored in traditional IP network, especially in the research area of web caching and CDN.

Web proxy caches reduce the volume of traffic between ISP and stub network by caching static Web content. In [7], cache schemes for achieving this goal are surveyed. Besides the cache schemes, the Web traffic pattern is studied by analyzing Web proxy traces by many research efforts such as the work presented by [4], in which the evidence and implications of Zipf-like distributions of Web object request are elaborated. Furthermore, some cooperative mechanisms among multiple Web proxy caches were proposed to exploit further the potential benefits of caching in contrast to the low cache hit rates due to independent proxies. These research works include [6] and [8].

In the research area of CDN, the problem of how to replicate objects optimally in terms of reducing the propagation delay and bandwidth consumption is studied extensively, including both mathematical (e.g. [21]) and practical efforts (e.g. [18]). In [21], Kangasharju et al. study the object replication problem with respect to minimizing the average hop number to get the object as this paper does. It assumes some mechanism to get the object from the nearest AS (corresponding to the node in this paper). In contrast, the caching problem of ICN discussed in this paper just assumes a path provided by the routing system to the original object.

VI. CONCLUSION

We formulate the in-network caching problem of ICN into Mixed-Integer Linear Programming problem. Via studying the properties of the optimal cache assignment and extensive simulations, we found that frequency-based cache policies like LFU perform well when considering minimizing the average hop number to get content. Through extensive simulations under various scenarios and configurations, we find that the performance gain brought by LB is limited in contrast to LFU, which implies that more sophisticated cache policies involving the distance factor should be considered to improve LFU. Our simulation results also show that, under a reasonable setting of cache size and request pattern, the average hop number to get content can be reduced significantly by nearly 50% in comparison to that of the case without in-network caching. Designing more intelligent cache policies considering distance factor and other topology information is the focus of our future work.

REFERENCES

[1] Bengt Ahlgren, Christian Dannewitz etc. A Survey of Information-Centric Networking (Draft), http://drops.dagstuhl.de/opus/volltexte/2011/2941/pdf/10492.KutscherDirk.Paper.2941.pdf.

[2] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, “Evaluating content management techniques for web proxy caches,” ACM SIGMETRICS Performance Evaluation Review, vol. 27, no. 4, 2000, pp. 3-11.

[3] S. Arianfar, P. Nikander, and J. Ott, “On content-centric router design and implications,” ACM ReArch 2010, November 30, 2010, Philadelphia, USA.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distributions: Evidence and implications,” IEEE INFOCOM, 1999.

[5] C. Williamson, “On filter effects in web caching hierarchies,” ACM Transactions on Internet Technology (TOIT), vol. 2, no. 1, 2002, pp. 47-77.

[6] R. Lancellotti, B. Ciciani, and M. Colajanni, “A Scalable Architecture for Cooperative Web Caching,” Web Engineering and Peer-to-Peer Computing, pp. 29-41.

[7] J. Wang, “A Survey of Web Caching Schemes for the Internet,” ACM SIGCOMM Computer Communication Review., vol. 29, no. 5, pp. 36-46,Oct. 1999.

[8] R. Tewari, M. Dahlin, H.M. Vin and J.S.Kay, “Design considerations for distributed caching on the Internet,” Proc. 19th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 273-284, June 1999.

[9] Rocketfuel, "An ISP Topology Mapping Engine," http://www.cs.washington.edu/research/networking/rocketfuel/

[10] PURSUIT, http://www.fp7-pursuit.eu/

[11] NDN, http://www.named-data.net/

[12] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, LIPSIN: Line Speed Publish/Subscribe Inter-networking," in Proceedings of the ACM SIGCOMM 2009 conference on Data communication. New York, NY, USA.

[13] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, A. Eriksson, et al. “Second netinf architecture description,” 4WARD EU FP7 Project, Deliverable D-6.2 v2.0, Apr. 2010, fP7-ICT-2007-1-216041-4WARD / D-6.2, http://www.4ward-project.eu/.

[14] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, M. Marchisio, I. Marsh, et al. “Design considerations for a network of information,” in Proceedings of ReArch'08: Re-Architecting the Internet, Madrid, Spain, Dec. 9, 2008.

[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in Proceedings of the 5th international conference on Emerging networking experiments and technologies, ser. CoNEXT '09. New York, NY, USA, 2009.

[16] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” in Proceedings of SIGCOMM'07, Kyoto, Japan, Aug. 27-31, 2007

[17] L. Muscariello, G. Carofiglio and M. Gallo, “Bandwidth and Storage Sharing Performance in Information Centric Networking,” Proceedings of the ACM SIGCOMM workshop on Information-centric networking, 2011, pp 26-31

[18] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K.K. Ramakrishnan, “Optimal content placement for a large-scale VoD system,” ACM CoNEXT 2010, Philadelphia, USA.

[19] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan and J.Wilcox, “Information-Centric Networking Seeing the Forest for the Trees,” Proceedings of the 10th ACM Workshop on Hot Topics in Networks, 2011.

[20] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, “Redundancy in network traffic: findings and implications,” SIGMETRICS/Performance’09, June 15–19, 2009, Seattle, WA USA.

[21] J. Kangasharju, J. Roberts, and K.W. Ross, “Object replication strategies in content distribution networks,” Computer Communications, vol. 25, no. 4, 2002, pp. 376-383.