Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking

Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie

Abstract—Content-centric networking (CCN) is a promising framework to rebuild the Internet's forwarding substrate around the concept of content. CCN advocates ubiquitous in-network caching to enhance content delivery, and thus each router has storage space to cache frequently requested content. In this work, we focus on the cache allocation problem, namely, how to distribute the cache capacity across routers under a constrained total storage budget for the network. We first formulate this problem as a content placement problem and obtain the optimal solution by a two-step method. We then propose a suboptimal heuristic method based on node centrality, which is more practical in dynamic networks with frequent content publishing. We investigate through simulations the factors that affect the optimal cache allocation, and perhaps more importantly we use a real-life Internet topology and video access logs from a large scale Internet video provider to evaluate the performance of various cache allocation methods. We observe that network topology and content popularity are two important factors that affect where exactly should cache capacity be placed. Further, the heuristic method comes with only a very limited performance penalty compared to the optimal allocation. Finally, using our findings, we provide recommendations for network operators on the best deployment of CCN caches capacity over routers.

Index Terms—Cache allocation, CCN, on-path, optimization, centrality

1 Introduction

Ubiquitous in-network caching of content is a key feature of content-centric networking (CCN) [1]. This architecture proposes to re-build the Internet's forwarding substrate around the concept of content, where nodes can issue *interest* packets that are forwarded (at layer-3) to sources that can, in return, serve *data* packets. To achieve this, each content chunk is allocated a globally unique identifier that all network entities, including routers, can comprehend. Through this, it becomes possible for any router to cache *data* packets and, subsequently, serve future requests via its cache, rather than forwarding *interests* to the origin of the content. Many benefits are expected, including improved performance [2], [3], lower network costs [4], and superior resilience.

Despite the elegance of this concept, there are a number of challenges to be addressed. Perhaps most notable is the potentially huge cost that would be involved in deploying ubiquitous cache storage on *every* router. In practice, with current technologies, this would be extremely expensive and energy intensive; for example, a CCN router with 10 TB of cache space using Flash-based solid-state drives (SSDs) would cost $300,000 and consume 500 W of power [5]. This leads us to conclude that, if deployments were achieved, a huge range of CCN router models would exist, with networks intelligently deploying high capacity ones only in the areas that need them the most. We therefore argue that one of the key challenges in CCN's future deployment is the *cache allocation problem*: given a finite set of cache storage and a specific topology, how should the storage be distributed across the CCN routers?

This problem has been studied extensively in other domains, with optimization techniques being used to calculate optimal cache allocation in multi-cache networks, such as CDNs [6], [7], web caching [8], [9] and IPTV [10], [11]. These, however, are based on specific applications and are only suitable for particular types of topologies (e.g., hierarchical, adaptive overlay structures). Instead, the constraints become far tighter in CCN, as caches are co-located with the (layer-3) infrastructure. Therefore, it is not possible to flexibly build your own (layer-7) cache topology as with traditional forms of cache allocation. This is further exacerbated by the fact that, practically speaking, only on-path caching is allowed, with *interest* packets always following FIB entries.

Past work has proven inconclusive in this domain, with a lack of consensus. For example, Rossi and Rossini [12] found that allocating more cache space in the "core" of the network would improve the performance of CCN when compared to homogeneous allocation. Subsequently, Psaras et al. [13] and Fayazbakhsh et al. [14] concluded the opposite: it would be better to allocate capacity at the edge. Indeed, one of the key challenges identified in CCN is precisely this: where exactly should cache capacity be placed [15], the core, the edge or a mix of both?

This paper seeks to address this gap, exploring the reasons behind some of these apparently conflicting results.

• *Y. Wang is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and the Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, CETC 54. E-mail: wangyonggong@ict.ac.cn.*
• *Z. Li and G. Xie are with Network Division, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China. E-mail: {zyli, xie}@ict.ac.cn.*
• *G. Tyson and S. Uhlig are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, United Kingdom. E-mail: {gareth.tyson, steve}@eecs.qmul.ac.uk.*

Manuscript received 23 June 2014; revised 10 Feb. 2015; accepted 11 Feb. 2015. Date of publication 8 Mar. 2015; date of current version 16 Dec. 2015. Recommended for acceptance by K. Li.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2409848

We correlate this problem with cache allocation and propose an optimal cache allocation algorithm to discover the key factors that impact the performance of caching in CCN. Furthermore, we propose a heuristic algorithm to compute a practically feasible and near-optimal cache allocation. We use both synthetic and real-life network topologies and content access logs to evaluate the impact of various factors and the performance of proposed algorithms. To summarize, we make the following contributions:

- We investigate the cache allocation problem in CCN, proposing an optimal solution for computing which CCN routers should be enabled with caching, and exactly how much storage should be allocated.
- We propose a practically feasible centrality-based heuristic method that does not depend on global content distribution information as required by the optimal solution. Our heuristic mechanism offers a key tool for assisting network operators in these decisions.
- We utilize our optimal solution to explore the key factors that impact the performance of cache allocation via extensive simulations. In particular, we perform the first comprehensive evaluation that considers the following factors: topology, network size, content popularity characteristics and object replacement strategies.
- We use real-life Internet topology and video content request logs to examine the performance of various cache allocation methods. The results reveal that compared with the exact optimal solution, the heuristic allocation comes with limited cache performance penalty. We also provide recommendations to network operators on the choosing of cache allocation policy.

While the exact optimal allocation and some preliminary results have been presented in [16], the heuristic mechanism and the experiments using a real-life Internet topology and video access logs are newly added in this extended version.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, a detailed model of CCN caching is presented, before proposing optimization techniques, both exact and heuristic. In Section 4, we perform simulations to explore the key factors that impact cache allocation and network performance. In Section 5, we bring together our findings to delineate a number of considerations. Finally, we conclude in Section 6.

2 Related Work

We classify related work into two groups: (1) cache allocation methods in CCN; and (2) optimization methods for current multi-cache networks.

2.1 Cache Allocation in CCN

A wealth of research has recently inspected the performance of caching in CCN, e.g., [2], [4], [17], [18], [19], [20], [21]. These works perform extensive simulation studies, providing a range of insights, including performance modeling [2], [17], deployment incentives [4], and cache routing [18]. All of them consider homogeneous deployments, where all routers have the same cache size. In practice, however, these homogeneous deployments are extremely unlikely. Instead, a variety of router types will exist, with networks deploying appropriately sized caches in strategic locations.

To our knowledge, there are only a few works that have considered heterogeneous cache allocation policies. Rossini and Rossi [12] were the first to study the cache allocation problem in CCN. They propose to deploy more cache space in the "core" routers of the network, rather than at the edge, and conclude that the gain brought by cache size heterogeneity is actually very limited. In contrast, a later work [13] concludes the opposite, finding that keeping larger caches at the edge is, in fact, more effective. Similarly, Fayazbakhsh et al. [14] questioned the necessity of ubiquitous caching and concluded that most of the performance benefits can be achieved by edge caching alone. We re-visit this apparent contradiction in Section 4. Our heuristic allocation differs from [12] in that we use the centrality metric for selecting the cache location. The cache quota allocated to routers is further computed using a greedy algorithm, rather than proportional to the centrality metric of routers as in [12].

Beyond the considerations of previous work, we also note that no prior research has considered factors outside the topology, e.g., the content popularity distribution. Our work brings perspective on the conclusions of previous work, as well as shedding light on various aspects of cache allocation.

2.2 Optimization Methods in Multi-Cache Network

The process of cache allocation is typically realized using content placement optimization algorithms, i.e., through solving the facility location problem. This has been extensively studied in different areas (e.g., the web), where two general approaches have been taken: Capacitated Facility Location (CFL) and un-capacitated facility location (UFL), where the capacity of a node is the maximum number of clients it can serve simultaneously, not the cache capacity.

Krishnan et al. [8] formulate en-route web caching as a standard UFL problem in a tree topology, and present solutions based on both dynamic programming and greedy heuristics, where the objective is to minimize the remaining traffic flow. Jiang and Bruck [9] also address coordinated en-route web caching. They show that this can be achieved without pre-fetching or pre-positioning by a central controller. However, the mentioned approaches treat the content files in the network as an un-splittable commodity. This assumption is not practical in CCN, where content is split into chunks.

In [6], a two-step algorithm is developed to solve the multi-commodity UFL problem with total capacity constraints in trees, and to provide approximations for general graphs. Similarly, [7] discusses the multi-commodity UFL problem, where each node has its own cache capacity constraint. The authors in [11] and [10] add the bandwidth of links as additional constraints in the contexts of CDNs and IPTV networks, respectively. Although these works are closely related to ours, it is not applicable to a CCN environment, which has certain unique constraints, namely: (1) exclusive use of on-path caching; (2) opportunistic caching, rather than pre-fetching; (3) unchangeable locations of available cache points, i.e., routers; and (4) diverse topologies that do not match predetermined templates. For example, most of the optimal solutions in CDNs [6], [7] or IPTV [10], [11] assume off-path caching, and rely on file pre-fetching. Similarly, all aforementioned works in web caching [8], [9] or more recent cloud caching [22], [23] employ fixed hierarchical tree topologies. It is important to study this issue in more generalized network topologies, oriented towards CCN's design, rather than in an application-specific context.

3 Optimal Cache Allocation in CCN

Before exploring the factors that impact cache allocation in CCN, it is necessary to devise a mechanism to compute their allocation optimally. Without this, it becomes impossible to differentiate between external impact factors (e.g., content popularity) and simply poor allocation decisions. This section delineates an exact optimization algorithm to solve the cache allocation problem, providing the upper bounds of performance. This optimality, however, comes at a high computational cost, making it infeasible for calculating large-scale and dynamic deployments. We thus expand this work by proposing a centrality-based heuristic method capable of handling Internet-scale topologies. We propose the former to enable an exact study of the impact of cache performance, whereas we propose the latter as a key tool for network operators to help inform cache deployments.

3.1 Cache Allocation Problem

A CCN network can be modeled as an undirected graph $G = (V, E)$, where V is the set of CCN routers with cache space, and E is the set of links in the network. Each content server/client is attached to a node in V. For simplicity, we do not distinguish the end-hosts from the CCN routers they are attached to. Due to the aggregation of hierarchical names in CCN, we also assume that, in most circumstances, at most one FIB entry will exist in each router for each content item, i.e., from the perspective of an individual client, only one destination exists for each object [24].

In our model, each content chunk, $f_i \in F$, therefore has a single node, $v_s \in V$, chosen as its origin server, denoted as $s(f_i) = v_s$. A content chunk is the minimum operating unit in cache management, and is called a cache entry in the rest of this paper. All content chunks are assumed to have the same size, which is normalized to be 1. When a request for f_i is sent from node v_t, $path(v_t, s(f_i)) = path(v_t, v_s) = \{v_t, \ldots, v_s\}$ denotes the path that the *interest* packet takes, where v_t and v_s are the client and server, respectively. Any intermediate router could satisfy the *interest* with a cached chunk. If node v_a is the first encountered router with a cached copy of the content, then the forwarding path $\{v_t, \ldots, v_s\}$ will be reduced to $\{v_t, \ldots, v_a\}$, and the length of the reduction $\{v_a, \ldots, v_s\}$ is defined as the benefit of caching f_i at v_a for node v_t, denoted as $b_{i,a,t}$. A simple example in Fig. 1 illustrates the definition of the benefit. Note that the cache benefit used in our optimal allocation algorithm is gauged through the traffic reduction via caching, which is measured by the reduction in hop count for *interest* packets (as in [4]). To calculate the benefit for a particular content, we multiply the length of the reduction $b_{i,a,t}$ with the probability, p_i, of the content f_i being requested.

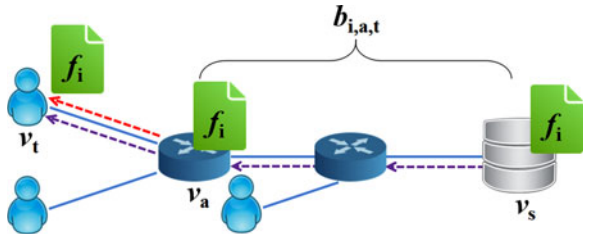


Fig. 1. Example illustrating the benefit of content placement. v_t and v_s denote the client and server of content f_i. The traffic flow corresponding to request f_i at v_t without caching is 3 (measured in hops). If f_i is cached at v_a, the traffic flow will be reduced to 1. Therefore, the benefit of caching f_i at v_a for node v_t is 2, $b_{i,a,t} = 2$.

As we measure the caching benefit by the reduction in hops taken by an *interest* packet, the objective of our optimization is to compute an allocation of cache capacity among a topology of routers such that the aggregate benefit is maximized (in line with previous optimization work [8], [9]). In other words, we aim to minimize the remaining traffic flow in the network. The key constraint is total cache capacity, as each network operator will have a finite amount of cache storage. The challenge is therefore to appropriately allocate this finite capacity across their infrastructure.

We formulate the optimal content placement problem in CCN as follows:

Maximize:

$$\sum_{f_i \in F} \sum_{v_t, v_a \in V} p_i \cdot x_{i,a} \cdot b_{i,a,t}. \tag{1}$$

Subject to:

$$\sum_{f_i \in F} p_i = 1 \tag{2}$$

$$x_{i,a} = \{0, 1\}, \forall f_i \in F, v_a \in V \tag{3}$$

$$\sum_{f_i \in F} \sum_{v_a \in V} x_{i,a} \leq c_{total}, \tag{4}$$

where $x_{i,a}$ is a binary variable taking the value 1 if content f_i is cached at node v_a. We assume that each content, f_i, is originally requested with the same probability p_i at each client. Eq. (4) states that the total cache space in the network is less than the constraint c_{total}. As the source of content f_i, $s(f_i)$, is determined by the content f_i uniquely, we do not need to consider this dimension in Eq. (1). The optimal content placement in CCN is therefore described by the set $x_{i,a}, f_i \in F, v_a \in V$ that maximizes the traffic saving. Finally, the optimal cache allocation can be calculated through $\sum_{f_i \in F} x_{i,a}, v_a \in V$.

3.2 Overview of Optimal Cache Allocation Algorithm

The objective function Eq. (1) can be rewritten as

$$max\left(\sum_{f_i \in F} \sum_{v_t, v_a \in V} p_i \cdot x_{i,a} \cdot b_{i,a,t}\right) \tag{5}$$

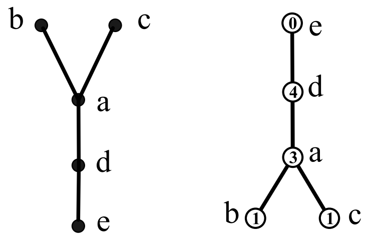


Fig. 2. The shortest path tree rooted at the server e. The number in the circle denotes the flow of the node.

$$= max\left(\sum_{f_i \in F} p_i \sum_{v_t, v_a \in V} x_{i,a} \cdot b_{i,a,t}\right) \tag{6}$$

$$= max\left(\sum_{f_i \in F} p_i \cdot b_i^{c_i}\right), \tag{7}$$

where $b_i^{c_i}$ is the benefit of allocating c_i cache entries[1] for content f_i across the whole network. Eq. (7) satisfies the standard knapsack problem formulation: how to get the largest benefit by allocating cache space for the set of content objects, F, where allocating c_i cache entries for f_i will provide benefit $p_i \cdot b_i^{c_i}$. Therefore, we formulate the optimized content placement problem in CCN as a general knapsack problem. As the probability of requesting content, p_i, is supposed to be known (a feasible method for obtaining p_i is described in [21]), $b_i^{c_i}$ in Eq. (7) is our main focus.

Considering that CCN exploits on-path caching, the benefits of caching not only depend on the cache locations ($X_i = \{x_{i,a}, v_a \in V\}$), but also on the location of the origin server ($s(f_i)$ for f_i). We assume that the *interest* packet always follows the shortest path to the server. Thus, $b_i^{c_i}$ in Eq. (7) is the benefit of X_i for all nodes on the shortest path tree (SPT) rooted at $s(f_i)$, implying that the mapping between content f_i and its original server $s(f_i)$ is an important input to the optimization.

Fig. 2 provides an example graph where node v_e serves file f_i. We define the flow of a node as the number of requests per time unit at that node, including the requests from the node itself as well as from its downstream nodes. We assume that each node in the tree will generate one request per time unit. Therefore, the total traffic flow without caching can be calculated by summing the flow at each node. The benefit of allocating $b_i^{c_i}$ is then equal to the total traffic flow for file f_i without caching minus the remaining flow with c_i cache entries. Table 1 lists the remaining flow and the benefit of allocating c_i cache entries with the optimal placement (obtained by the method proposed in [8]).

As Eq. (7) satisfies the standard knapsack problem formulation, the content placement problem in CCN can be divided into two sub-problems: (1) the cache location problem in the SPT; and (2) the knapsack problem to solve the whole of Eq. (7). The former problem has been solved in [25] as a k-means problem with $O(cn^2)$ complexity, where c is the number of cache nodes and n is the total number of nodes in the graph. Typically, the latter is solved through dynamic programming. As the k-means problem in trees has been proven to be piecewise linear non-decreasing concave [26], the knapsack of different contents can be solved optimally through a greedy method.

1. A cache entry is a fixed-size item of cache space in a router, equal to the size of one *data* chunk.

TABLE 1
The Benefit of Cache Allocation for f_i ($b_i^{c_i}$)

c_i (locations)	Remaining flow	Benefit $b_i^{c_i}$
0	9	0
1 (a)	3	6
2 (a, d)	2	7
3 (a, d, b)	1	8
4 (a, d, b, c)	0	9

3.3 Optimal Cache Algorithm Description

Algorithm 1 provides an outline of the exact optimization algorithm used to compute the optimal cache allocation. The notations are summarized in Table 2.

In Algorithm 1, the first step computes the benefit of cache placement on the SPT rooted at each server $v_s \in s(F)$; the second step computes the incremental benefit of the cth ($0 < c < n$) cache entry for each content; the third step allocates the cache space by choosing the largest benefit increment in F greedily; the last step maps X using C and $Y^{a,c}$, where $C = \{c_i\}, f_i \in F$. It is noteworthy that the algorithm does not require the exact popularity of a content item, but only the popularity distribution of all content items.

The output of Algorithm 1, X, is a $N \times n$ binary array describing the optimal content placement in the network with c_{total} cache entries. Finally, the sum of the columns (or rows) of X, $\sum_{f_i \in F} x_{i,a}, v_a \in V$ ($\sum_{v_a \in V} x_{i,a}, f_i \in F$), can be considered as the optimal cache allocation across nodes (or contents).

The complexity of the above algorithm is mostly determined by steps 1 and 3. By using the max-heap data structure, the greedy algorithm in step 3 can be implemented at a complexity of $O(c_{total} logN)$, where N is the number of content items. Given that the complexity of the cache

TABLE 2
Summary of Notations

Notation	Meaning
V	Set of nodes
E	Set of edges
F	Set of content chunks
n	$n = \|V\|$
N	$N = \|F\|$
v_a	Node $a, v_a \in V, a \leq n$
f_i	ith popular content,$f_i \in F, i \leq N$
$s(f_i)$	Server of content f_i
p_i	Probability that content f_i is requested
c_i	Cache capacity allocated for content f_i
$x_{i,a}$	Binary variable indicating cache f_i on node v_a
$y_b^{a,c}$	Binary variable indicating node v_b is one of the c optimal locations in the SPT rooted at v_a
$b_{i,a,t}$	Benefit of caching f_i on node v_a for node v_t
$b_i^{c_i}$	Benefit of caching f_i with c_i entries
bs_a^c	Benefit of c optimal entries on SPT rooted at v_a

location problem in the SPT is $O(sn^3)$ in step 1, the overall complexity of our optimization method is $max(O(sn^3), O(c_{total}logN)), s \leq n, c_{total} < nN$, where $s = |s(F)|$ is the number of servers. Although this optimal approach would not scale to Internet-wide deployments, it provides a reference to explore cache performance in CCN.

Algorithm 1. Exact optimal cache allocation algorithm.

```
                                        ▷ Step 1
 1: for all $v_s, v_s \in s(F)$ do
 2:     for all $c, 0 < c < n$ do
 3:         Compute $bs_s^c$ and $y_b^{s,c}$ as in [25]
 4:         $Y^{s,c} \leftarrow \{y_b^{s,c}\}, v_b \in V$
 5:     end for
 6: end for
                                        ▷ Step 2
 7: for all $c, 0 < c < n$ do
 8:     $b_i^c \leftarrow bs_a^c, f_i \in F, v_a = s(f_i)$
 9:     $\Delta b_i^c \leftarrow b_i^c - b_i^{c-1}, f_i \in F$
10: end for
                                        ▷ Step 3
11: $c_i \leftarrow 0, \forall f_i \in F$
12: while $\sum_{f_i \in F} c_i < c_{total}$ do
13:     $i \leftarrow \arg\max_{j|f_j \in F}(\Delta b_j^{c_j} \cdot p_j)$
14:     $c_i \leftarrow c_i + 1$
15: end while
                                        ▷ Step 4
16: $x_{i,a} \leftarrow y_a^{s(f_i),c_i}, \forall v_a \in V, f_i \in F$
17: $X = \{x_{i,a} | f_i \in F, v_a \in V\}$
```

3.4 Heuristic Cache Allocation Algorithm

The solution from the previous section, although optimal, is not practically relevant for a dynamic environment where the publishing or the deletion of content is so frequent that it is costly to maintain global information. We thus further propose a heuristic method suitable for highly dynamic networks. The most important differences from the exact method include:

- We use a heuristic method to obtain the approximate optimal cache locations in the SPT, rather than the dynamic programming in the exact method. We prefer the top centrality nodes as the cache locations, which dramatically reduces the computing complexity of finding the cache location.
- We do not require the exact location information of content servers, but make the assumption that it follows a certain kind of distribution. Without loss of generality, we assume that each node in the network has the same probability to be chosen as the origin server of any content.

Our heuristic is based on the observation that the optimal cache location in SPT trees highly overlaps with the top centrality metric nodes, irrespective of the root node location. In other words, no matter where the original server for content is located, the top centrality nodes are systematically good candidates for caching it. We provide an explanation for this. First, there is a small compact core in any scale-free network, from which almost all other nodes are within a distance of $loglog(n)$ [27]. The core is composed by the nodes

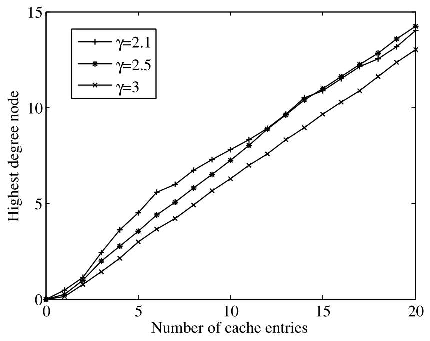


Fig. 3. Overlap between the optimal cache locations and the top degree nodes (n = 1,000).

whose degree is larger than $n^{1/loglog(n)}$, which is proved to be almost fully meshed. During the SPT building process (from a certain root node), one of the core nodes will be reached in few hops first, then most descendants will be reached from the high-degree neighbors of the core node(s). Given the very small diameter of scale-free networks (in the order of $log(n)/loglog(n)$), the generated SPT will be flat with few high fan-out nodes in it.

Next, we recall that the benefit of caching on a node is calculated by the flow of the node multiplied by the distance to the SPT root. Since the distance to the SPT root is less than the diameter of the topology, which is relatively small comparing with the variety of node flow, the benefit of caching on a node is dominated by the traffic passing through it. Obviously, two types of nodes have a high amount of traffic on the SPT: 1) the high fan-out nodes, and 2) the nodes between high fan-out nodes and the root. Consistent with [8], which found little performance loss using the greedy method compared to exact dynamic programming, we also choose the cache location by finding the largest traffic nodes greedily. Given that core nodes are fully-meshed, almost all neighbors of the core node in the SPT are not part of the SPT, meaning that caching at one of these core nodes will not affect the traffic of the others. Therefore, when choosing the optimal cache locations on any SPT with the greedy method, the core nodes with higher fan-out and more traffic will be appropriate candidates.

Fig. 3 shows the overlap between the top degree nodes and the optimal cache locations. Let DC_i denote the top i degree centrality (DC) nodes in the network; $OPT_{i,j}$ denotes the i optimal cache locations on the SPT rooted at node j. The overlap of top degree nodes and the optimal locations of the SPT rooted at node j is $DC_i \cap OPT_{i,j}$; the overlap ratio of the SPT rooted at node j with i cache entries is $|DC_i \cap OPT_{i,j}|/i$. In Fig. 3, the x-axis is the number of cache entries allocated to a certain content, and the y-axis is the expectation of the overlap ratio if its original server is randomly chosen among all the network nodes uniformly. Here, we rely on three scale-free topologies generated by the BA model [28], with different degree distribution parameters. We observe that about $3/4$ of the optimal cache locations coincide with the top degree nodes, especially when the scale-free exponent is small. We therefore claim that the top degree nodes are good candidates for the optimal cache locations.

Based on the above observations, Algorithm 2 provides a description of our heuristic algorithm in CCN. In the algorithm, bsh_a^c denotes the benefit of allocating cache entries on the top c centrality nodes in the SPT rooted at node v_a. The first step computes the centrality of each node; the second step calculates the benefit of allocating cache entries to the top c centrality nodes, bs^c, $0 < c < n$; the third step computes the incremental benefit of the cth $(0 < c < n)$ cache entry for each content; the fourth step allocates the cache space by choosing the largest benefit increment in F greedily; the last step maps X using C and node centrality, where $C = \{c_i\}, f_i \in F$.

Algorithm 2. Heuristic Cache Allocation Algorithm.

```
                                                    ▷ Step 1
 1: for all $v_a, v_a \in V$ do
 2:     Compute the centrality of $v_a$
 3: end for
                                                    ▷ Step 2
 4: for all $v_a, v_a \in V$ do
 5:     for all $c, 0 < c < n$ do
 6:         Select top $c$ centrality nodes as caching locations
 7:         Compute $bsh_a^c$
 8:     end for
 9: end for
                                                    ▷ Step 3
10: for all $c, 0 < c < n$ do
11: $bsh^c \leftarrow \sum_{v_a \in V} bsh_a^c / n$
12: end for
13: for all $c, 1 < c < n$ do
14:     $\Delta bsh^c \leftarrow bsh^c - bsh^{c-1}$
15: end for
                                                    ▷ Step 4
16  $c_i \leftarrow 0, f_i \in F$
17  while $\sum_{f_i \in F} c_i < c_{total}$ do
18:     $i \leftarrow \arg\max_{j | f_j \in F}(\Delta bsh^{c_j} \cdot p_j)$
19:     $c_i \leftarrow c_i + 1$
20: end while
                                                    ▷ Step 5
21: for all $f_i, f_i \in F$ do
22:     $S \leftarrow$ set of top $c_i$ centrality nodes
23:     for all $v_j, v_j \in S$ do
24:         $x_{i,j} \leftarrow 1$
25: end for
26: end for
27: $X = \{x_{i,j} | v_j \in V, f_i \in F\}$
```

Similar to the exact method, when the heuristic content placement X is obtained, the corresponding cache allocation can be easily calculated by summing all content placements at each node. However, unlike the optimization method, X in Algorithm 2 finds the best content caching in a statistical sense. In other words, in any network where each content has one fixed origin server, some distributed cache replacement policies may beat the exact pre-fetching based on X. Therefore, we suggest to use the right cache replacement strategy together with the heuristic cache allocation in highly dynamic networks. From the experiments of Section 4, we find that the heuristic allocation with the LFU cache replacement strategy provides comparable performance to the optimal allocation with pre-fetching.

The computational complexity of the heuristic is decided by the most time consuming steps: step 2 and step 4. In step 2, the complexity of calculating the benefit of a SPT bsh_a^c, $0 < c < n$ is $O(n)$, $n = |V|$. Thus, the complexity of computing bsh^c, $0 < c < n$ is $O(n^2)$. The complexity of step 4 is the same as in Algorithm 1, $O(nNlog(N))$, $N = |F|$. Therefore, the overall complexity of the heuristic is $max(O(n^2), O(nNlog(N)))$. Given that the node degree is our default centrality metric, we assume in this paper that the complexity of computing centrality is less than $O(nNlog(N))$.

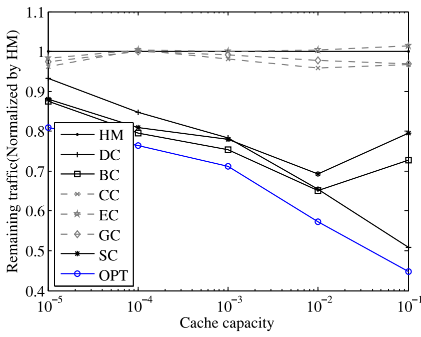
4 Analysis of Cache Allocation

This section explores the key factors that impact cache allocation, as well as how accurately our heuristic can inform these allocations. As of yet, there is no consensus on what factors should be considered in CCN cache allocation. There is not even consensus, broadly speaking, on where caches should be allocated: the core, the edge, or a combination of both [12], [13], [14], [15]. As such, using our optimal algorithm (OPT), we begin by exploring the types of deployment that could benefit from caching, and how capacity should be allocated among them. Then, we evaluate how our heuristic performs compared to the optimal allocation.

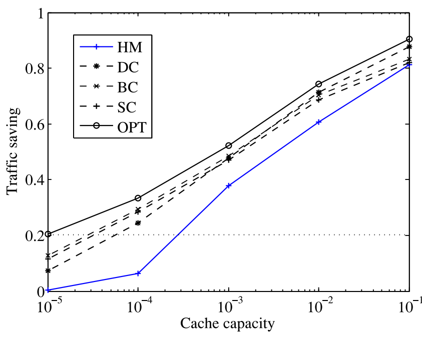
4.1 Experiment Setup

We have developed a discrete event based simulator that models caching behavior in various graph structures. Similarly to [4], we are primarily interested in the reduction in the hop count, and therefore we do not model traffic congestion or processing delay as in [14]. Although our lightweight simulator cannot capture the data plane features as done by, e.g., ndnSIM [29] or ccnSim [30], it manages to handle topologies with thousands of nodes, which is critical for our experiments. Configuring the simulator consists of topology generation and request pattern generation.

To study a range of configured topologies, we rely on synthetic generation. This allows us to control various properties of interest, e.g., the degree distribution and clustering. Here, we focus on power-law networks, as Internet-like graphs have similar features [31]. To generalize our results more, we also utilize two different flavors of power-law graph [31]. First, we employ the Barabási-Albert (BA) model [28] to emulate topologies with a few high centrality nodes (default $\gamma = 2.5$ and $m = 2$); this gives insight into an "ideal" caching scenario (i.e., high request overlap). However, it does not necessarily best model the real world and, therefore, we also utilize the Watts-Strogatz (WS) small-world topology model [32] to capture clustering. This results in more realistic topologies (e.g., an ISP network [33]) that allow us to better map our implications to current network deployments. In this model, first, each node v_i is assigned an expected degree κ_i, which is actually obtained from the BA model in our experiment. Then, all nodes are uniformly distributed on a ring space. Finally, links are set among nodes according to the expected degree and the metric distance on the ring. For example, the probability that link $\{v_i, v_j\}$ exists is proportional to $\kappa_i \kappa_j / d_{i,j}^{\alpha}$. A higher α value will create more clustering. By default, each generated topology consists of 1,000 router nodes. Once we generate the topology, we attach 100 servers, sharing 10,000 objects. We randomly choose their individual points of attachment,



(a) Remaining traffic normalized by homogeneous allocation performance.



(b) Traffic saving of cache allocation.

Fig. 4. Caching performance of different allocation methods. HM: homogenous allocation; DC: Degree Centrality; BC: Betweenness Centrality; CC: Closeness Centrality; EC: Eccentricity Centrality; GC: Graph Centrality; SC: Stress Centrality; OPT: optimal cache allocation.

before distributing the objects across the sources uniformly. During each time interval, all objects are requested by each node separately using a Zipf distribution: $\sum_{i=1}^{N}(c/i^{\beta}) = 1$ ($\beta = 1$).[2] Besides, by default, the global cache capacity is set as $c_{total} = 1\%$ which is normalized by nN. This allows us to capture the impact of both the number of nodes *and* the number of content files. Finally, we use a measured network topology and a real-life content access logs for evaluation.

4.2 Importance of Cache Allocation

Before investigating the key factors that impact cache allocation in CCN, we investigate the importance of cache allocation itself. More specifically, we would like to confirm that optimal cache allocation has a direct impact on performance, as opposed to cache capacity alone. We carry out simulations with our optimal algorithm (OPT, Algorithm 1), alongside several other cache allocation schemes. The cache replacement policy used in all tests is LFU, except the optimized method which depends on global knowledge and pre-fetching.

Fig. 4 presents the cache performance of different algorithms. The x-axis presents the global cache capacity (c_{total}), which is normalized by the number of nodes multiplied by the number of content files (i.e., nN). We measure the performance by the fraction of traffic that remains in the network, shown on the y-axis. To improve comparability, we also normalize the remaining traffic in Fig. 4a as a fraction of the homogeneous allocation.

2. Although typical, we do not use α to denote the skew factor, due to its previous use in parameterizing the WS graph.

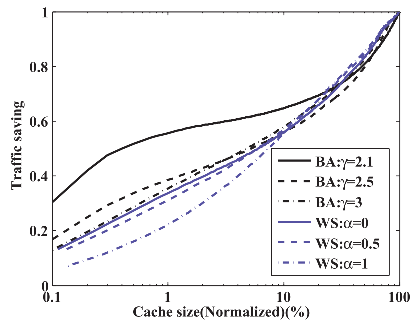
The heuristic allocations based on closeness centrality (CC), graph centrality (GC) and eccentricity centrality (EC) taken from [12] contribute little to the cache performance. They provide less than 5 percent in traffic reduction. In fact, EC actually results in an increase in traffic, due to its negative correlation with other centrality metrics [12]. In contrast, the other three metrics, degree centrality, betweenness centrality and stress centrality (SC), offer far greater traffic saving. Interestingly, among the three useful centrality based allocations, BC and SC are more effective when the total cache space is relatively small, whereas DC has better performance when the total budget is more than 1 percent. We find that BC and SC never allocate cache space to stub nodes, regardless of the overall capacity budget. This policy is reasonable when the total budget is small and all the cache resources are obtained by the core nodes, but it is inefficient when there is spare cache capacity that could be placed on the home gateways provided to users. None of these existing algorithms approach the gains made by the optimal algorithm.

Fig. 4b also shows the total benefit, measured by the fraction of traffic removed via caching. Note that we do not show CC, EC and GC due to their poor performance. We see that with increased cache capacity, the benefits increase logarithmically. We observe that the homogeneous allocation cannot benefit from this logarithmic law and performs quite poorly when the total cache budget is small. For example, to achieve a traffic reduction of 20 percent, the homogeneous allocation will require over 10 times more cache space compared to the optimal allocation. This highlights the importance of an appropriate cache allocation scheme, as opposed to cache size alone. We therefore confirm that cache placement *does* have a significant impact on network performance in CCN. Further, we have also shown that our optimized algorithm offers far superior results to the existing state-of-the-art.

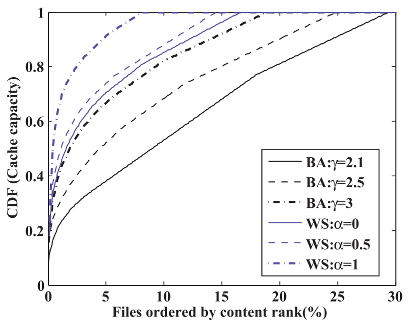
4.3 Impact of the Topology Structure

Next, we seek to discover if the core or the edge is the most appropriate place for caches. We define the nodes with a high centrality metric value as the “core” and the nodes with low centrality metric value as the “edge”. Using the topology models detailed in Section 4.1, we create a variety of network topologies (controlled by the parameters γ and α) and compute the optimal cache placements using the OPT algorithm. The larger the value of γ, the heavier the power-law, meaning that the network is more strongly hierarchical, resulting in a few highly central nodes. As α increases, the clustering in the network also increases. In contrast, small values of α correspond to network topologies more similar to random networks.

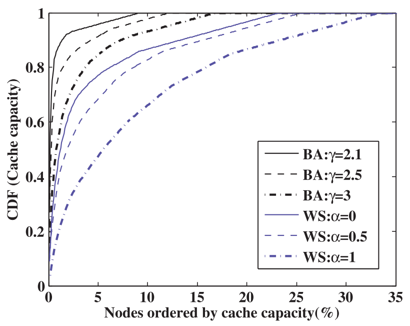
Fig. 5a presents the distribution of traffic savings across various cache sizes in the network. We observe that the BA topology does consistently better with small cache capacities when compared to WS. The reason for this behavior is that a BA topology will tend to have central nodes that aggregate a lot of requests, and therefore benefit more from caching. For example, the BA topology ($\gamma = 2.1$) can achieve over 30 percent traffic reduction with a cache size of just 0.1 percent. This can be compared against the WS topology which achieves less than 20 percent for an equivalent cache



(a) Benefit of caching in optimal location



(b) Cache allocation of content files



(c) Cache allocation over nodes

Fig. 5. Impact of topology properties using BA and WS models.

capacity. The explanation for this behavior is that the WS topology has more path diversity, and therefore benefits less from aggregation. As a consequence, in a WS topology, more cache locations must be used across the network, especially towards the edge, which decreases the cache efficiency as popular content will be replicated across multiple caches. This trend becomes even worse for larger values of α in the WS model (more clustering), resulting in even lower caching performance. Despite the lower traffic savings, we believe that placing caches at the edge is still desirable for smaller networks that do not benefit from large levels of aggregation.

We next investigate how the content is distributed across the caches. Ideally, one would wish to reduce the level of redundancy in regions of the network, to improve cache utilization (assuming that performance, rather than resilience, is the aim). Fig. 5b presents the cumulative distribution function (CDF) of the distribution of content in the caches; for example, it shows that the majority of cache capacity is allocated to the most popular object (file index 0). More interestingly, we see that the topology has a notable impact on how capacity is allocated to each file. We find that the WS topology (particularly when $\alpha = 1$) allocates a large fraction of the capacity to the top few objects. This happens because it is impossible to aggregate large portions of the requests to caches, as there is no single "core" to operate as a prominent caching point. Instead, by pushing caches towards the edge, it becomes necessary to fill each cache with the same popular objects.

Finally, Fig. 5c presents how the capacity is distributed across all routers in the network. Lower node indexes indicate nodes that are closer to the core. Once again, we see a similar trend to Fig. 5b. With more decentralized topologies, it becomes necessary to spread the cache capacity across the network towards the edge, as highlighted by the WS models. In contrast, the BA topologies result in far more centralized cache placement, where content is primarily stored in the core, i.e., skewed towards lower node indexes. This suggests that different deployments could require entirely different allocation approaches, and therefore that a one-size-fits-all approach would be inappropriate. It is important to remember that here we discuss a global network with a "core". In practice, the Internet has been seen to move away from a hierarchical topology towards a flatter and more similar topology to WS than BA [34]. As such, not only does optimality differ between different networks, but it will also differ over time across the whole Internet.

4.4 Impact of Network Size

We now investigate how the number of nodes in the network impacts placement in Fig. 6. The homogeneous allocation consistently achieves low performance. For instance, when $n = 2k$, the optimal allocation with $c = 0.001$ achieves the same traffic savings as the homogeneous allocation with 10 times more cache capacity. We also see that the benefit of the optimal allocation is proportional to the network size, n, while the homogeneous allocation is actually quite insensitive to the network size.

We also note another important repercussion of this finding in relation to [12], which concluded that the heterogeneous allocation brings little improvement compared with the homogeneous allocation. Initially, this seemed to be in stark contrast to our findings. However, closer inspection highlighted that the experiments in [12] were only based on topologies with 68 nodes or fewer. Instead, it appears from our results that the benefits of the heterogeneous allocation only become apparent with larger topologies. Considering that CCN is expected to target large networks or even an Internet-scale deployment, we expect that heterogeneous cache placements will be highly relevant. Either way, this shows that as the deployment of CCN increases in scale, the need for appropriate cache allocation will become increasingly important.

4.5 Impact of Content Popularity

To investigate the impact of content popularity distribution, Fig. 7 presents the impact that differing levels of skew have on performance. Specifically, we vary the skew parameter, β, where the probability that the ith object

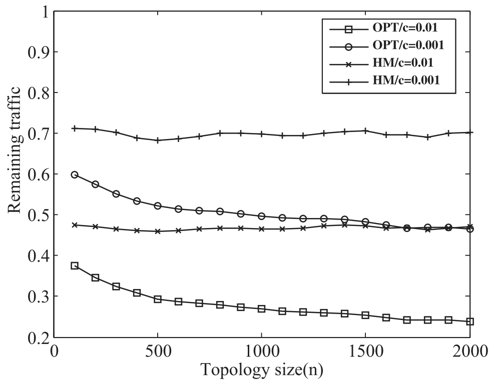


Fig. 6. The remaining traffic in different networks sizes, from 100 to $2{,}000$ nodes.

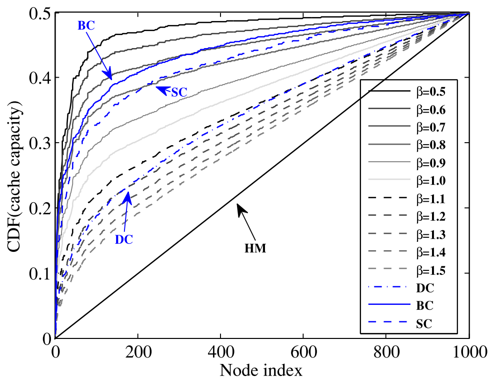


Fig. 7. The optimal allocation of different content popularity. β is the Zipf skew parameter.

being requested is proportional to $1/i^{\beta}$. The experiments are performed using the BA topology. The node index along the x-axis refers to the attachment order in the BA model; lower values indicate nodes in the core also with a larger centrality metric value.

Fig. 7 shows that a less skewed popularity distribution (i.e., smaller β) results in more cache space allocated in the core. Indeed, it then becomes necessary to consolidate cache capacity across a larger number of consumers to ensure sufficient interest overlap to generate cache hits. This means that highly skewed popularity request patterns (e.g., YouTube [35], mobile VoD system [36]) will be better served by edge caching, while more uniform popularity distributions (e.g., catch-up TV [37]) would be better served by core caching.

Another important observation is that as β increases, the performance of the optimal allocation begins to converge towards that of homogeneous allocation. We can see that as the range of objects being requested becomes more skewed, evenly distributing the caches towards the edge of the network becomes more effective. This allows each stub network to serve its own consumers effectively, without requiring the aggregation of consumers offered by in-core caching. This latter point has the additional advantage of reducing transit traffic, as well as delay. Optimal cache allocation might therefore be suited to help ISPs reduce their transit costs.

On the other hand, the allocations in proportion to degree centralit, betweenness centrality and stress centrality are only suitable for certain content popularity. In other words, besides the total cache budget (Fig. 4), content popularity is another key factor that impacts the performance of centrality-based cache allocation.

Another important factor is the *number* of objects. Fig. 8 plots the amount of traffic saving in the optimal and homogeneous allocations, where the cache capacity, normalized by nN, is 1 percent. Through this, the total cache capacity in the network increases with the volume of content files N. We see that greater savings (i.e., less remaining traffic) can be achieved as the number of objects increases in proportion to the cache capacity. This occurs because the number of objects *requested* increases at a sub-linear rate compared to the number of objects themselves, allowing the caches to consolidate their capacity better. One could say that this is optimistic; as such, Fig. 8 also presents the results when maintaining a constant total cache capacity (100 entries per node on average). In this case, the traffic saving decreases approximately logarithmically with the number of content items in both allocations.

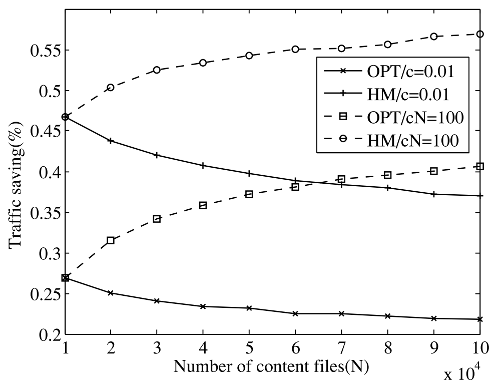


Fig. 8. The effect of the number of content files on the traffic saving. The dash lines show the traffic saving where the cache capacity is increased with the content volume. The solid lines show the traffic saving where the total cache capacity is fixed.

Consequently, we conclude that the increasing scale of the content will, indeed, need to be matched by increasing the scale of caching resources. However, the superior logarithmic decrease in performance suggests that significant effort should be spent on cache allocation deployment policies, in an attempt to mitigate the costs of the constant cache expansion.

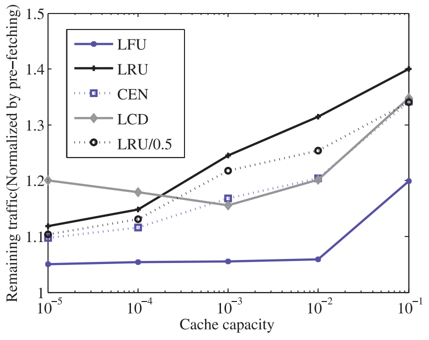
4.6 Impact of Cache Replacement Policy

As the theoretical optimal allocation algorithm in this paper is obtained using optimal object placement, it requires oracle-based pre-fetching, which is, of course, impractical in a real-world context. Consequently, we evaluate the performance of different well-known replacement strategies with the optimal allocation.

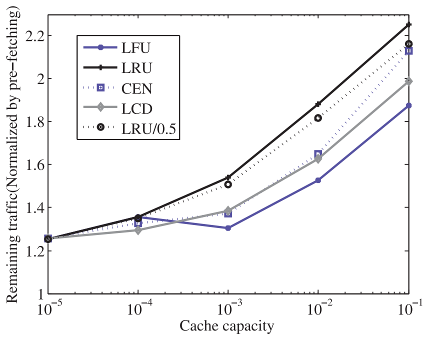
To illustrate the relative cache performance compared to the pre-fetching method (exact optimal placement), the remaining traffic on the y-axis in Fig. 9 is normalized using the one achieved by the optimal placement. Besides the default cache replacement strategy (LFU), we also experiment with several other algorithms: (1) Least Recently Used (LRU) with "cache everything everywhere" [1]; (2) LRU with the centrality-based caching policy [19] (CEN); (3) LRU with LCD caching policy [38]; and (4) LRU with fixed caching probability [19] (LRU$/p$,$p = 0.5$).

Fig. 9 shows that the caching performance of the optimal allocation is much better than the homogeneous allocation for all replacement strategies. Therefore, the benefit of heterogeneous optimal cache placement does *not* require oracle-based pre-fetching: significant gains can be made with various simple cache replacement strategies.

We also find that the allocation of cache resources impacts the performance of these replacement methods. For example, with a homogeneous cache allocation, LCD is found to be the best with low cache capacities, confirming the findings of [20]. However, with the optimal cache allocation, it does worst at these low capacities. In this scenario, nearly all the cache space in the network is allocated to a few core nodes. Therefore, there are not enough cache nodes to form the multi-hop cache paths as LCD requires (unlike in the homogeneous allocation).

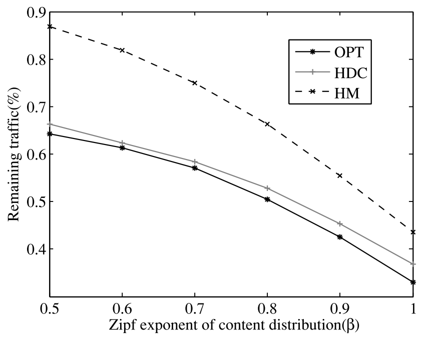


(a) Optimal allocation.

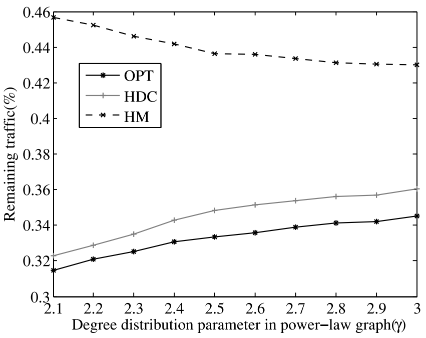


(b) Homogeneous allocation.

Fig. 9. Caching performance of different replacement strategies. The cache performance of replacement strategies is normalized by optimal oracle-based pre-fetching.



(a) Varying content skewness.



(b) Varying topologies.

Fig. 10. Cache performance of heuristic compared to optimal and homogeneous allocations. Node degree is used as the heuristic centrality metric. There are 256 nodes with 64 content servers in this scenario.

Similar variability is also observed in LFU, which has the best performance in most scenarios, apart from the homogeneous allocation when the total cache budget is less than 0.01 percent. We find that the cache space allocated to each node is so small in the homogeneous allocation that it cannot estimate the probability of arriving content properly. In this situation, reducing the cache replacement times (as CEN and LCD do) or avoiding the amplification of replacement errors (as LCD) is helpful to improve performance. This shows that there is no one-size-fits-all replacement strategy for different cache allocations. Instead, network operators will need to choose the most suitable replacement strategy according to their network environment as well as the cache allocation method in use.

4.7 Cache Performance of Heuristic Allocation

We next confirm that our heuristic allocation can offer equivalent results without the high computational costs. Fig. 10 compares the performance of the heuristic mechanism (HDC) against both the optimal allocation (OPT) and the homogeneous allocation (HM). We choose to inspect performance while varying the popularity skew and the degree distribution, as they are the most important practical metrics.

Fig. 10a shows that the heuristic allocation offers superior performance to simple homogeneous allocation across all popularity skews. More importantly, the heuristic approach closely approximates optimality consistently, regardless of popularity skew. In fact, the performance penalty is bounded by only 4 percent on average. As such, even the worst-case performance of the heuristic algorithm can offer significant improvements. Fig. 10b also presents the caching performance of the heuristic allocation with different topologies, which are generated using the BA model. Once again, the heuristic method shows close performance to the optimal cache allocation throughout. Interestingly, it also exhibits superior performance in more hierarchical topologies, where node degree centrality displays higher variability. Despite these variations, the performance penalty for different topologies is always below 5 percent.

The results confirm that the heuristic algorithm can achieve near optimality, outperforming traditional homogeneous allocations. As such, we argue that our centrality based heuristic scheme is a powerful tool for network planners, particularly those who wish to shape their deployments more dynamically. Unlike the optimal solution, the heuristic approach also does not require the specific server location information.

4.8 Cache Performance Based on Measured Datasets

The above experiments use synthetic topologies and content request profiles, which allow us to examine the cache performance in various experimental scenarios. To have an understanding of our optimal allocation algorithm in a practical situation, we further use a real-life Internet topology and a collection of access logs from a commercial Internet video provider. The topology is obtained from [39], providing the connectivity between cities from the China Telecom network. The graph is made of 321 nodes, 1,507 links, has an average path length of 2.84 hops, and an average degree of 9.39. The video access logs are obtained from [40], which were collected from PPTV,[3] a leading video

3. http://www.pptv.com/

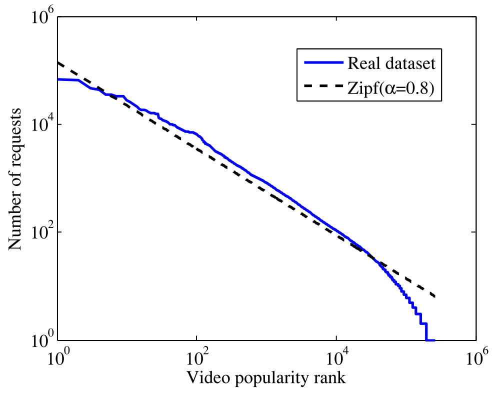


Fig. 11. Video popularity follows the Zipf distribution.

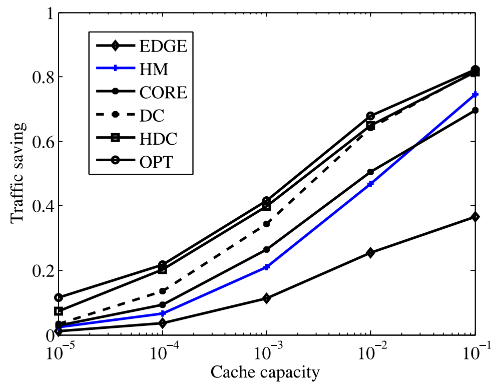


Fig. 12. The cache performance of different allocations using real dataset.

provider in China, for two weeks at the end of 2011. In total, we obtained approximately 7.9 million requests across 264 k videos. Fig. 11 shows the video popularity derived from the access logs in log-log scale. The video popularity roughly follows a Zipf distribution with $\beta = 0.8$. The geographical locations (at city-level) of clients of individual requests are available in the video access logs. We use such location information to map the source of each request to the corresponding node in the topology. The original severs of individual videos, on the other hand, are attached to the nodes that are selected uniformly at random.

Fig. 12 shows the performance of different cache allocation algorithms, including optimal allocation (OPT), heuristic allocation based on degree centrality (HDC), and other four algorithms based only on topological properties: proportional allocation by degree centrality, homogeneous allocation (HM), homogeneous allocation only on core nodes (CORE) and homogeneous allocation only on edge nodes (EDGE). As expected, the optimal allocation obtains the best cache performance. Our heuristic allocation based on centrality performs close to the optical allocation with less than a 4 percent performance penalty.

Of the four allocation algorithms based on topology properties, DC achieves the best caching performance. Nevertheless, its performance can only approach the optimal allocation when the cache capacity grows to as large as 10 percent. Recall that our centrality based heuristic allocation differs from DC in that the heuristic algorithm uses the content popularity distribution information to allocate cache space to nodes with high centrality values, while DC directly allocates cache space to nodes in proportional to the centrality values. In other words, in the case that the content popularity information is available, our heuristic allocation is the best choice. Otherwise, if such information is not available (e.g., when severing a particular type of contents for the first time), allocation in proportion to node degree is a better choice.

5 Summary of Findings

The previous section has explored the key factors that impact optimal cache allocation. Clearly, our results cannot be directly applied to a specific practical deployment, but we can draw a number of general conclusions from our study:

- Allocating cache capacity across the network in a homogeneous manner is highly suboptimal. Instead, capacity should be allocated in a heterogeneous manner. The benefits of this, however, only become apparent with larger networks (e.g., >100 nodes). We argue that CCN router design should therefore ensure that cache capacity is easily pluggable and extensible.
- The topology has a significant impact on the optimal cache placement. In inter-AS type topologies (i.e., similar to BA), cache capacity should be pushed into the core as much as possible. This allows requests to be effectively aggregated, improving hit rates, due to the high level of *interest* path co-location. In contrast, ISP-type networks (i.e., similar to WS) should distribute capacity in a more homogeneous manner, pushing storage towards the customer edge due to a lack of a well-defined core.
- The type of content popularity handled by the network will alter the optimal deployment. Demand that is more uniformly distributed is better handled by pushing caches into the core (e.g., an Internet Exchange Point [41]) to better aggregate requests. For highly skewed demands, caches must be pushed to the edge.
- As the number of objects increases, the importance of strategic cache placement also increases. Homogeneous allocation strategies do substantially worse, while the performance of the optimal allocation decreases in an approximately logarithmic manner.
- The benefit of heterogeneous optimal cache placement does *not* require oracle-based pre-fetching. Significant gains can be made with simple cache replacement strategies. Furthermore, the cache replacement strategy can have a notable impact on performance.
- Compared with the exact optimal solution, the heuristic allocation comes with limited cache performance penalty. Given that the heuristic allocation is independent of the server information for content items, it is suited to highly dynamic networks where content is frequently published. This makes the approach a valuable tool for any network planners wishing to deploy CCN infrastructure.
- In practice, network operators should make the decision of cache allocation based on the available

information of network topology and content access patterns. In case that both network topology properties and content access patterns are available, optimal cache allocation or centrality based heuristic allocation are preferred. However, in case the content popularity distribution information is unavailable, allocation proportional to node degree is a better choice.

6 Conclusion

This work has presented an exact optimization method to find the optimal cache allocation in CCN. We have explored many factors that affect cache placement, and how they subsequently impact performance (measured by traffic reduction). We also proposed a centrality-based heuristic for dynamic networks with frequent content publishing. Our experiments, using both synthetic and real-life traces, have shown that the benefits of heterogeneous cache allocation are significant. We have found that many aspects may affect cache performance, including topological characteristics and content request patterns. This highlights that a one-size-fits-all approach is very unlikely to be optimal for CCN's deployment. Instead, network operators must make the decision of cache allocation based on their network topology and content access patterns.

There are several directions for future work. First, our work relies on the aggregated content popularity distribution. However, content might exhibit different temporal dynamics, which would possibly affect the performance of cache replacement policies. It is thus wise to investigate how the temporal dynamics impact the cache performance using both the optimal and heuristics allocation algorithms. Second, although Internet video services will be the dominant applications in term of traffic, the future Internet will also host other types of contents like web content and cloud storage. As such, it is also important to analyze the cache allocation with other forms of content.

Acknowledgments

This work was supported by the National Basic Research Program of China Nos. 2012CB315801, the National Natural Science Foundation of China Nos. 61133015 and 61272473, and the National High-tech R&D Program of China No. 2013AA013501. Yonggong's work was also supported by the Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, the 54th Research Institute of China Electronics Technology Group Corporation. Z. Li is the corresponding author.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. ACM 5th Int. Conf. Emerging Netw. Experiments Technol.*, 2009, pp. 1–12.

[2] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," in *Proc. 10th Int. IFIP TC 6 Conf. Netw.*, 2011, pp. 78–91.

[3] Q. Wu, Z. Li, J. Zhou, H. Jiang, Z. Hu, Y. Liu, and G. Xie, "Sofia: Toward service-oriented information centric networking," *IEEE Netw.*, vol. 28, no. 3, pp. 12–18, May 2014.

[4] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *Proc. IEEE Int. Conf. Comput. Commun. Netw.*, 2012, pp. 1–7.

[5] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. ACM SIGCOMM Workshop Inf.-Centric Netw.*, 2011, pp. 44–49.

[6] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis,"Joint object placement and node dimensioning for internet content distribution," *Inf. Process. Lett.*, vol. 89, no. 6, pp. 273–279, 2004.

[7] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakiss, "On the optimization of storage capacity allocation for content distribution," *Comput. Netw.*, vol. 47, no. 3, pp. 409–428, 2005.

[8] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[9] A. Jiang and J. Bruck, "Optimal content placement for en-route web caching," in *Proc. IEEE 2nd Int. Symp. Netw. Comput. Appl.*, 2003, pp. 9–16.

[10] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proc. ACM 6th Int. Conf. Emerging Netw. Experiments Technol.*, 2010, p. 4.

[11] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[12] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proc. IEEE INFOCOM NOMEN*, 2012, pp. 280–285.

[13] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2920–2931, Nov. 2014.

[14] S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 147–158.

[15] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: Seeing the forest for the trees," in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, p. 1.

[16] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *Proc. IEEE Int. Conf. Netw. Protocol*, 2013, pp. 1–10.

[17] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[18] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: Hamlet-like doubts in ICN," in *Proc. 2nd Edition ICN Workshop Inf.-Centric Netw.*, 2012, pp. 7–12.

[19] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *Proc. 11th IFIP Netw.*, 2012, pp. 27–40.

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

[21] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proc. 8th ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2012, pp. 15–26.

[22] Z. Li, Y. Huang, G. Liu, F. Wang, Z.-L. Zhang, and Y. Dai, "Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices," in *Proc. 22nd Int. Workshop Netw. Operating Syst. Support Digital Audio Video*, 2012, pp. 33–38.

[23] Y. Huang, Z. Li, G. Liu, and Y. Dai, "Cloud download: Using cloud utilities to achieve high-quality content distribution for unpopular videos," in *Proc. ACM 19th Int. Conf. Multimedia*, 2011, pp. 213–222.

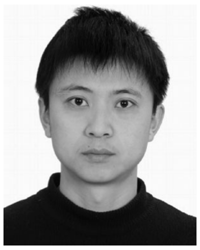
[24] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, "A survey on content-oriented networking for efficient content delivery," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 121–127, Mar. 2011.

[25] A. Tamir, "An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs," *Oper. Res. Lett.*, vol. 19, no. 2, pp. 59–64, 1996.

[26] R. Shah, "Faster algorithms for k-median problem on trees with smaller heights," Computer Science Technical Reports, Paper 1579, Report Number: 03-030, 2003, http://docs.lib.purdue.edu/cstech/1579

[27] F. Chung and L. Lu, "The average distances in random graphs with given expected degrees," *Nat. Acad. Sci.*, vol. 99, no. 25, pp. 15879–15882, 2002.
[28] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
[29] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: Ndn simulator for ns-3," NDN, Tech. Rep. NDN-0005, 2012.
[30] The ccnsim homepage [Online]. Available: http://perso.telecom-paristech.fr/ drossi/index.php?n=Software.ccnSim, 2012.
[31] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: Degree-based vs. structural," in *Proc. ACM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2002, pp. 147–159.
[32] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 409–410, 1998.
[33] B. Quoitin, V. V. den Schrieck, P. Francois, and O. Bonaventure, "IGen: Generation of router-level internet topologies through network design heuristics," in *Proc. 21st Int. Teletraffic Congr.*, 2009, pp. 1–8.
[34] D. Fay, H. Haddadi, A. Thomason, A. W. Moore, R. Mortier, A. Jamakovic, S. Uhlig, and M. Rio, "Weighted spectral distribution for internet topology analysis: Theory and applications," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 164–176, Feb. 2010.
[35] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1357–1370, Oct. 2009.
[36] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, and G. Peng, "Watching videos from everywhere: A study of the PPTV mobile VOD system," in *Proc. ACM Conf. Internet Meas. Conf.*, 2012, pp. 185–198.
[37] H. Abrahamsson and M. Nordmark, "Program popularity and viewer behaviour in a large tv-on-demand system," in *Proc. ACM Conf. Internet Meas. Conf.*, 2012, pp. 199–210.
[38] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Perform. Eval.*, vol. 63, no. 7, pp. 609–634, 2006.
[39] Y. Tian, R. Dey, Y. Liu, and K. W. Ross, "China's internet: Topology mapping and geolocating," in *Proc. INFOCOM*, 2012, pp. 2531–2535.
[40] Q. Wu, Z. Li, and G. Xie, "Codingcache: Multipath-aware CCN cache with network coding," in *Proc. ACM SIGCOMM Workshop Inf.-Centric Netw.*, 2013, pp. 41–42.
[41] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large european IXP," in *Proc. ACM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 163–174.

Yonggong Wang received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently with the Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, 54th Research Institute of China Electronics Technology Group Corporation. His research interests include content centric networking and Internet of thing.



Zhenyu Li received the BS degree from Nankai University in 2003 and the PhD degree in computer science from ICT, CAS in 2009. He is an associate professor at the Institute of Computing Technology (ICT), Chinese Academy Sciences (CAS). His research interests include future Internet and Internet measurement.

Gareth Tyson received the PhD degree from Lancaster University in 2010. He is a lecturer at Queen Mary University of London. His research interests include Internet measurements, content distribution, and the future Internet.

Steve Uhlig received the PhD degree from the University of Louvain in 2004. He is a professor of networks at Queen Mary University of London. His research interests include large-scale Internet measurements, software-defined networking and the behavior of Internet routing and traffic.

Gaogang Xie received the PhD degree in computer science from Hunan University, in 2002. He is a professor in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include future Internet architecture, programmable virtual router platform, Internet measurement and modeling.