ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/256078717

Optimal Cache Allocation for Content-Centric Networking

Conference Paper · October 2013

DOI: 10.1109/ICNP.2013.6733577

CITATIONS
101

READS
270

5 authors, including:

Yonggong Wang
Chinese Academy of Sciences
5 PUBLICATIONS **173** CITATIONS

SEE PROFILE

Zhenyu Li
Chinese Academy of Sciences
106 PUBLICATIONS **1,490** CITATIONS

SEE PROFILE

Gareth Tyson
Queen Mary, University of London
160 PUBLICATIONS **2,057** CITATIONS

SEE PROFILE

Steve Uhlig
Queen Mary, University of London
210 PUBLICATIONS **8,379** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

INET TU Berlin View project

Privacy-Preserving Decentralized Abuse Detection View project

All content following this page was uploaded by Gareth Tyson on 16 May 2014.

The user has requested enhancement of the downloaded file.

Optimal Cache Allocation for Content-Centric Networking

Yonggong Wang[1], Zhenyu Li[1], Gareth Tyson[2], Steve Uhlig[2], and Gaogang Xie[1]

[1]Institute of Computing Technology, Chinese Academy of Sciences, {wangyonggong, zyli, xie}@ict.ac.cn
[2]Queen Mary, University of London, {gareth.tyson, steve}@eecs.qmul.ac.uk

***Abstract*—Content-Centric Networking (CCN) is a promising framework for evolving the current network architecture, advocating ubiquitous in-network caching to enhance content delivery. Consequently, in CCN, 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 formulate this problem as a content placement problem and obtain the exact optimal solution by a two-step method. Through simulations, we use this algorithm to investigate the factors that affect the optimal cache allocation in CCN, such as the network topology and the popularity of content. We find that a highly heterogeneous topology tends to put most of the capacity over a few central nodes. On the other hand, heterogeneous content popularity has the opposite effect, by spreading capacity across far more nodes. Using our findings, we make observations on how network operators could best deploy CCN caches capacity.**

I. 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], lower network costs [3], 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 [4]. 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 fixed 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 [5], [6], Web caching [7], [8] and IPTV [9], [10]. 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 becomes 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 [11] found that allocating more cache space in the "core" of the network would improve the performance of CCN when compared to homogeneous allocation. After that, Psaras *et al.* [12] and Fayazbakhsh *et al.* [13] 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 [14], the core, the edge or a mix of both?

This paper seeks to address this gap, exploring the reasons behind some of these conflicting results. More generally, we aim to discover the key factors that impact the performance of caching in CCN. We correlate these findings with cache allocation, highlighting how an intelligent allocation policy can dramatically increase performance. We show that a one-size-fits-all approach would be entirely inappropriate and that, instead, different networks must make measured decisions based on their individual operating conditions. To summarize, we make the following key 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 should be allocated.
- We utilize our solution to explore the key factors that impact the performance of cache allocation via extensive simulations. We perform the first comprehensive evaluation that considers the following factors: topology, network size, content popularity characteristics and ob-

ject replacement strategies. To achieve this, we exploit multiple synthetic topology models that capture different types of network to characterize the types of deployments suitable for each. In conjunction, we utilize various popularity distributions and replacement algorithms to similarly understand their interactions.
- We provide an analysis of the key considerations that network operators should take into account when deciding their own cache allocation policy.

The remainder of the paper is organized as follows. In Section II, we discuss related work. In Section III, a detailed model of CCN caching is presented, before employing optimization techniques to resolve the cache allocation problem in CCN. In Section IV, we perform simulations to explore the key factors that impact cache allocation and network performance. Following this, in Section V, we bring together our findings to delineate a number of considerations that should be made by network operators during CCN's deployment. Finally, we conclude the paper in Section VI.

II. Related Work

We classify related work into two groups: (1) cache allocation methods in CCN; and (2) optimization methods for current multi-cache networks, such as CDNs, the Web and IPTV.

A. Cache allocation in CCN

A wealth of recent research has inspected the performance of caching in CCN, e.g., [15], [16], [17], [3], [18], [2], [19]. These works perform extensive simulation studies, providing a range of insights, including performance modeling [2], deployment incentives [3] and cache routing [16]. All of them however consider homogeneous deployments, where all routers have the same cache size. In practice, however, these homogeneous deployments are extremely unlikely. Instead, a wealth of router types will exist, with networks deploying appropriately sized caches in strategic locations.

We know of only three works that have considered heterogeneous cache allocation policies. Rossi and Rossini [11] 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. They use several metrics to measure the centrality of routers, including degree, closeness and betweenness. Through this, they allocate cache capacity proportionally to the centrality metric of a router, and conclude that the gain brought by cache size heterogeneity is actually very limited. In contrast, a later work [12] concludes the opposite, finding that keeping larger caches at the edge is, in fact, more effective. Similarly, Fayazbakhsh *et al.* [13] 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 IV.

Beyond the considerations of the 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.

B. Optimization methods in multi-cache network

The process of cache allocation is typically realized using content placement optimization algorithms. In other words, it solves the Facility Location Problem. This has been extensively studied in alternate domains (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 means the maximum number of clients it can serve simultaneously, not the cache capacity.

Krishnan *et al.* [7] 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 [8] also address coordinated en-route web caching. Perhaps most importantly, they show that this can be achieved without pre-fetching or pre-positioning by a central controller. However, all of the mentioned approaches treat the content files in the network as an un-splittable commodity. In contrast, this is not practical in CCN, which separates content into chunks.

In [5], 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, [6] discusses the multi-commodity UFL problem, where each node has its own cache capacity constraint. [10] and [9] add the bandwidth of links as additional constraints in the contexts of CDNs and IPTV networks, respectively. Although this work is closely related, 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 [5], [6] or IPTV [9], [10] assume off-path caching, and rely on file pre-fetching. Similarly, all aforementioned work in web caching [7], [8] employ fixed hierarchical tree topologies, that simply would not exist in CCN. It is therefore important to study this issue in more generalized network topologies, oriented towards CCN's design, rather than in an application-specific context.

III. 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. Note that we do not intend this approach for large-scale use by real network operators but, instead, exploit it later as a basis for our evaluation.

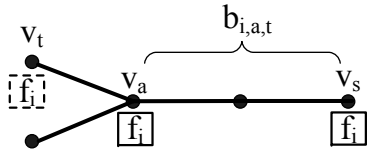


Fig. 1. Example illustrating the benefit of content placement. v_t and v_s denote the client and server of content f_i. The traffic flow corresponding to request f_i at v_t without caching is 3 (measured by hops). If f_i is cached at v_a, the traffic flow will be reduced to 1. Therefore, the benefit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

A. Cache allocation problem

A CCN network can be modeled as a 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 can 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 [20].

In our model, each content chunk, $f_i \in F$, will therefore have 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, $path(v_t, s(f_i)) = path(v_t, v_s) = \{v_t, \cdots, v_s\}$ 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 $\{v_t, \cdots, v_s\}$ will be reduced to $\{v_t, \cdots, v_a\}$, and the length of the reduction $\{v_a, \cdots, v_s\}$ 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 measured through the traffic reduction via caching, which is measured by the reduction in hop count for *interest* packets (as in [3]). 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.

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 [7], [8]). 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 finite accessibility to 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:

Notation	Meaning
V	Set of nodes
E	Set of edges
F	Set of contents
n	$n = \|V\|$
N	$N = \|F\|$
v_a	Node $a, v_a \in V, a \leq n$
f_i	i-th 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
b_i	Benefit of caching f_i with X_i placement
$b_{i,a}$	Benefit of caching f_i on node v_a for all nodes
$b_{i,a,t}$	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 deploying c cache entries on SPT rooted at v_a
α	Clustering control parameter in WS model
β	Zipf distribution exponent
γ	Degree distribution parameter in scale-free topology

TABLE I
SUMMARY OF NOTATIONS.

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. $b_{i,a,t}$ is the benefit of caching f_i at v_a for client v_t. 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 $\sum_{f_i \in F} x_{i,a}, v_a \in V$.

B. Overview of Optimal Cache Allocation Algorithm

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

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

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

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

where $b_i^{c_i}$ is the benefit of allocating c_i cache entries[1] for content f_i. Eq. (7) satisfies the standard knapsack problem

[1]A cache entry is a fixed-size item of cache space in a router (equal to the size of one *data* packet).

File f_i	Cache size c_i	Remaining flow	Benefit $b_i^{c_i}$
f_5	0	9	0
f_5	1	3	6
f_5	2	2	7
f_5	3	1	8
f_5	4	0	9

TABLE II
THE BENEFIT OF CACHE ALLOCATION FOR A GIVEN CONTENT FILE ($b_i^{c_i}$). THE SPT ROOTED AT v_e IS CHOSEN AS AN EXAMPLE (THE RIGHTMOST TREE IN FIG. 2).

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, f_i, is supposed to be known, $b_i^{c_i}$ in Eq. (7) is our main focus in the following algorithm.

Considering that CCN exploits on-path caching, the benefits of caching do not only depend on the cache locations ($X_i = x_{i,a}, v_a \in V$) but also 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 and its SPTs rooted at different content servers. In Fig. 2, we assume that there are 5 content files in this graph, and each content file is served at one node. In this example, we assign node v_e as the server of file f_5. We define the flow of a node as the number of requests per time unit at that node. 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 remaining flow and the benefit of allocating c_5 cache entries with the optimal placement (obtained by the method proposed in [7]) are listed in Table II.

The content placement problem in CCN can be divided into two sub-problems: (1) the cache location problem in the SPT to obtain $b_i^{c_i}$ in Eq. (7); and (2) the knapsack problem to solve the whole of Eq. (7). The former problem has been solved in [7], 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 [21], the knapsack of different contents in Eq. (4) can be solved optimally through a greedy method.

C. Algorithm Description

We now provide an outline of the exact optimization algorithm used to compute the optimal cache allocation:

1) Compute the benefit of cache placement on the SPT rooted at each server,v_a,$v_a \in s(F)$.
   a) Compute the benefit of deploying c cache entries on the SPT rooted at v_a, $bs_a^c, c = \{0, \cdots, n-1\}$ following the dynamic programming method described in [7], [22];
   b) Save the optimal cache location $Y^{a,c}$ obtained from the above dynamic programming for future use, where $Y^{a,c} = \{y_b^{a,c}\}, v_b \in V$, $y_b^{a,c}$ is a binary value that indicates whether node v_b is one of the c optimal locations in the SPT rooted at v_a.
2) Compute the incremental benefit of the c^{th} cache entry for content f_i, denoted as Δb_i^c.
   a) $b_i^c = bs_a^c, f_i \in F, v_a = s(f_i), c = \{0, \cdots n-1\}$.
   b) $\Delta b_i^c = b_i^c - b_i^{c-1}, f_i \in F, c = \{1, \cdots n-1\}$
3) Allocate the cache space by choosing the largest benefit increment in F greedily.
   a) Initially, $c_i = 0, f_i \in F$,$b_{opt} = 0$, where c_i denotes the number of cache entries allocated for content f_i, and b_{opt} is the total benefit of $\sum_{f_i \in F} c_i$ cache entries;
   b) $c_i = c_i + 1$, $\Delta b_i^{c_i} \cdot p_i = max\{\Delta b_j^{c_j} \cdot p_j, f_j \in F\}$, where p_i is the probability of requesting content f_i;
   c) $b_{opt} = b_{opt} + \Delta b_i^{c_i}$;
   d) if $\sum_{f_i \in F} c_i < c_{total}$, go to b);else, iteration stop.
4) Map X using C and $Y^{a,c}$, where $C, C = \{c_i\}, f_i \in F$.
   a) $X = \{x_{i,a} = y_a^{s(f_i),c_i}\}\, f_i \in F, v_a \in V$, where $X_{i,a}$ is a binary variable indicating that cache f_i on node v_a

The output of this algorithm, 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 seen as the optimal cache allocation across nodes (or contents).

As the $k-means$ problem in trees has been proven to be piecewise linear non-decreasing concave [21], the greedy solution is also optimal. In other words, the knapsack problem can be solved greedily. The complexity of the above algorithm is mostly determined by steps 2 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 contents. Given that the complexity of the cache location problem in the SPT is $O(sn^3)$ in step 2, 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 an appropriate basis to explore the cache performance in CCN.

IV. ANALYSIS OF CACHE ALLOCATION

This section explores the key factors that impact cache allocation. 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 [11],

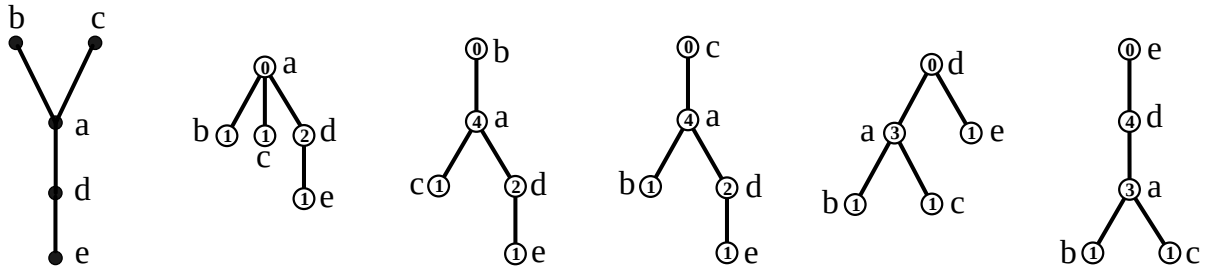


Fig. 2. The shortest path trees (SPTs) rooted at different content servers. The number in the circle denotes the flow of the node, which is defined as the number of requests for content per time unit at that node, including the requests from the node itself as well as from its downstream nodes. Each node in the network generates one request per time unit.

[12], [14], [13]. Therefore, we utilize our optimal algorithm to understand the types of deployment that could benefit from caching, and how capacity should be allocated among them.

A. Experiment setup

First, we detail the methodology taken to achieve our goals. We have developed a discrete event based simulator that models caching behavior in various graph structures. As with [3], we are primarily interested in reducing hop counts, and therefore we do not model traffic congestion or processing delay as done in [13]. Although our lightweight simulator cannot capture the data plane features as done by, e.g., ndnSIM [23] or ccnSim [24], it manages to handle topologies with thousands of nodes, which is critical for our experiments. Configuring the simulator consists of topology generation and request patterns generation, which we discuss in the following paragraphs.

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 [25], as Internet-like graphs have similar features [26]. To generalize our results more, we also utilize two different flavors of power-law graph [27]. 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 [29] to capture clustering. This results in more realistic topologies (e.g., an ISP network [30]) that allows us to better map our implications to current network deployments. In this model, firstly, 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, before distributing the objects across the sources uniformly. For simplicity, we do not distinguish the end hosts from the router nodes they are attached to. 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] Finally, by default, the global cache capacity is set as $c_{total} = 1\%$ which is normalized by nN.

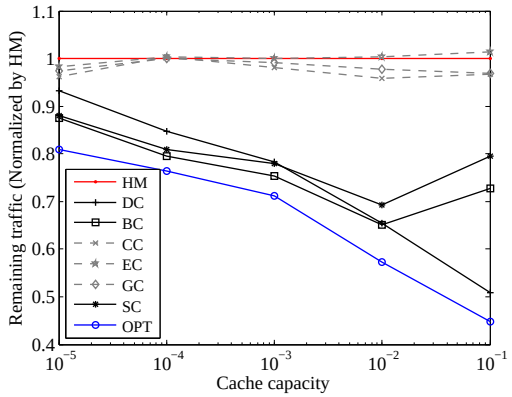
B. Importance of Cache Allocation

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. To achieve this, we perform simulations of our optimal algorithm (OPT), alongside several other cache allocation schemes. We therefore recreate the algorithms detailed in [11], as well as our own. The cache replacement policy used in all tests is LFU, except the optimized method which depends on global knowledge and pre-fetching. Details about the impact of the cache replacement policy will be explored later.

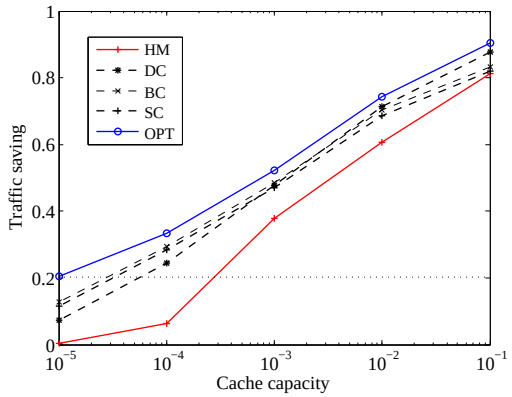
Fig. 3 presents the cache performance of the different algorithms using the aforementioned default simulation parameters. The x-axis presents the global cache capacity (c_{total}), which is normalized by the number of nodes multiplied by the number of content files (i.e., nN). This allows us to capture the impact of both the number of nodes *and* the number of content files. We measure the performance by the fraction of traffic that remains in the network, shown on the y-axis. If, for example, $c_{total} = 1$ (100%), then the fraction of remaining traffic will be 0. To improve comparability, we also normalize the remaining traffic in Fig. 3(a) as a fraction of the homogeneous allocation.

The heuristic allocations based on closeness centrality (CC), graph centrality (GC) and eccentricity centrality (EC) contribute little to the cache performance. All achieve well below 5% in reductions. In fact, EC actually results in an increase in traffic, due to the negative correlation coefficient value with other centrality metrics in [11]. In contrast, the other three

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



(a) Remaining traffic normalized by homogeneous allocation performance.



(b) Traffic saving of cache allocation.

Fig. 3. 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.

metrics, degree centrality (DC), betweenness centrality (BC) 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%. We find that the BC/SC allocation methods never allocate cache space to stub nodes, regardless of the overall capacity budget. This policy might be 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. As expected, none of these existing algorithms approach the gains made by the optimal algorithm (OPT).

Fig. 3(b) 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%, the homogeneous allocation will require over 10 times more cache space compared to the optimal allocation. This highlights the importance of a proper cache allocation scheme, as opposed to cache size alone. We therefore confirm that cache placement *does* have a significant impact on network performance. Further, we have also shown that our optimized algorithm offers far superior results to the existing state-of-the-art.

C. Edge vs. core: impact of the topology structure

Here, we seek to understand the impact of the topology on cache allocation. More generally, we seek to discover if the core or the edge is the most appropriate place for caches to be situated. We define the nodes with 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 IV-A, 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 close to random networks.

Fig. 4(a) presents the distribution of traffic savings across various cache sizes in the network. For example, it shows that, for the BA topology ($\gamma = 2.5$), at least 20% of traffic is saved with a 1% cache size. 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% reductions with a cache size of just 0.1%. This can be compared against the WS topology which gets well below 20% for an equivalent cache capacity. This occurs because the WS topology has more path diversity, and therefore benefits less from aggregation. As a consequence, in a WS topology, more caches must be placed across the network, especially towards the edge, which also decreases the cache efficiency (popular content will be replicated in many caches). This trend becomes even worse for larger values of α in the WS model (more clusters), resulting in even lower 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. 4(b) 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. 4(c) presents how the capacity is distributed across all routers in the network. Lower nodes indexes indicate nodes that are closer to the core. Once again, we see a similar trend to that in Fig. 4(b). 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 (skewed towards lower node indexes). This suggests that different deployments could require entirely different allocation approaches, suggesting that a one-size-fits-all approach would be completely inappropriate.

D. Impact of network size

So far, we have seen that the topological structure of the network can have a significant impact on the optimal cache placement. As of yet, however, we have only dealt with a single network size (1k). We now investigate how the number of nodes in the network impacts placement. Fig. 5 illustrates the effect of the network size, n, in CCN, where both homogeneous and optimal allocation are evaluated. Note that we present both because the homogeneous allocation is the de-facto consideration in most prior work, while the optimal allocation is the upper-end benchmark.

Once again, we can see that the homogeneous allocation is suboptimal, consistently achieving 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. Specifically, in the homogeneous experiments, the traffic savings remain fairly constant for all network sizes. This suggests that network providers who demand predictable savings rather than optimal savings may actually prefer the simplicity of such a deployment.

We also note another important repercussion of this finding in relation to [11], 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 [11] were only based on topologies with 68 nodes or fewer. Instead, it appears from our results, that the benefits of the heterogeneous cache allocation only become apparent with larger topologies. Considering that CCN is targeted at larger networks or even an Internet-scale deployment, it seems likely that heterogeneous cache placements would be highly relevant.

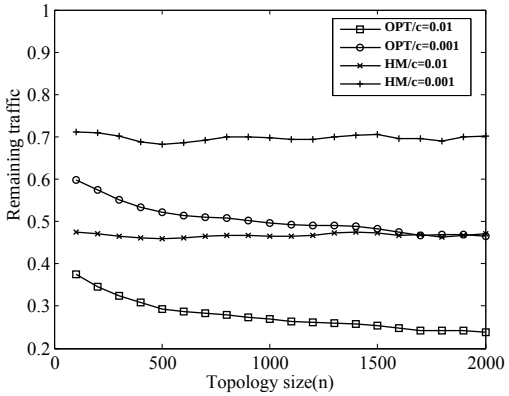


Fig. 5. The remaining traffic in different sized networks, from 100 to 2,000 nodes.

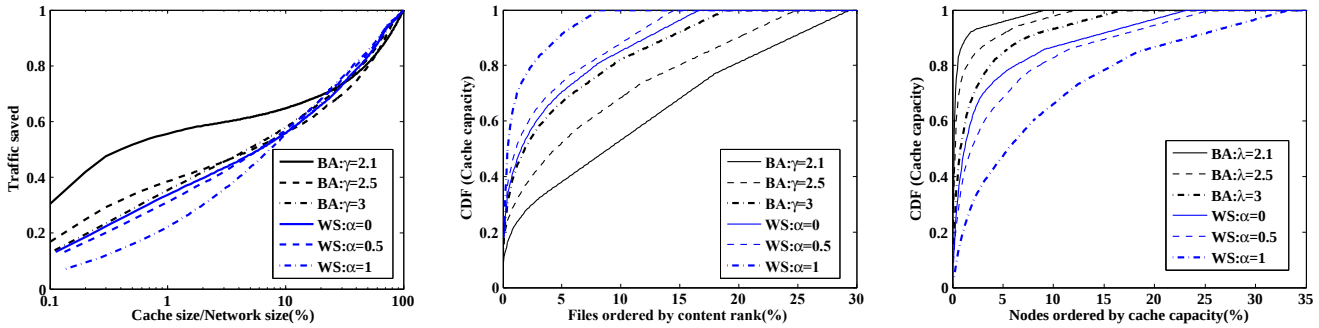
E. Impact of content popularity

Caching, of course, relies heavily on the skewed distribution of popularity among objects, and CCN is no different. To investigate this, Fig. 6 presents the impact that differing levels of skew have on the performance (note that for completeness we also include DC, BC, SC placement schemes, as well as optimal). Specifically, we vary the skew parameter, β, where the probability that the i^{th} object being requested is proportional to $1/i^{\beta}$. 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 large centrality metric value. The experiments are performed using the BA topology.

Fig. 6 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 between a larger number of consumers to ensure sufficient interest overlap to generate cache hits. For example, an edge cache serving 10 consumers is unlikely to generate any hits for unpopular objects, as there would not be enough requests. In contrast, a core cache serving 100k consumers is far more likely to find subsequent requests for unpopular objects. This means that highly skewed popularity request patterns (e.g., YouTube [31], mobile VoD system [32] or Vimeo [33]) will be better served by edge caching, while more uniform popularity distributions (e.g., catch-up TV [34]) 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 the homogeneous allocation. We can see that as the range of objects being requested becomes more skewed, it becomes more effective to evenly distribute the caches towards the edge of the network. 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 further advantage of reducing transit traffic, as well as delay. Optimal cache allocation might therefore be suited to help ISPs reduce their transit costs.

Whereas the popularity distribution of objects has a notable



(a) Benefit of caching in optimal location (b) Cache allocation of content files (c) Cache allocation over nodes

Fig. 4. Impact of topology properties using BA and WS models.

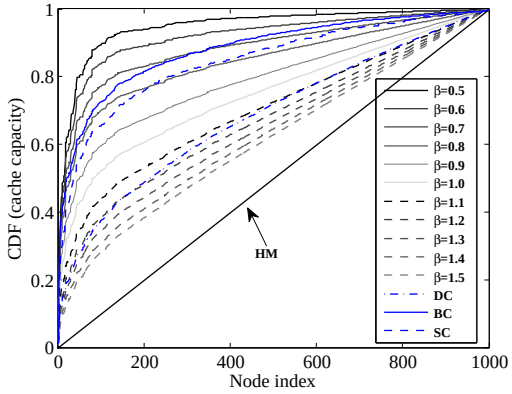


Fig. 6. The optimal allocation of different content popularity. β is the Zipf skew parameter. Heuristic methods based on degree centrality (DC), betweenness centrality (BC) and stress centrality (SC) are also shown in blue lines (note β does not change their allocations).

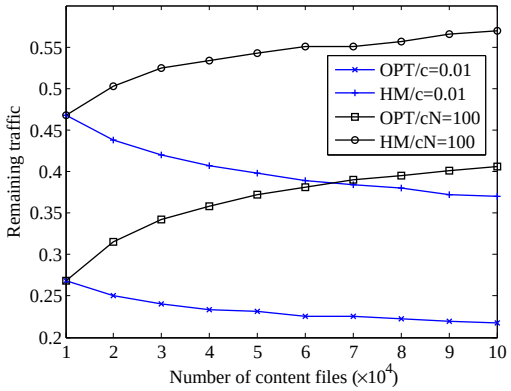


Fig. 7. The effect of the number of content files ($10^4 < N < 10^5$) on the remaining traffic. The blue lines show the remaining traffic where the cache capacity is increased with the content volume. The black lines show the remaining traffic where the total cache capacity is fixed (100 cache entries per node on average).

impact, a further important factor is the *number* of objects. Fig. 7 plots the amount of remaining traffic in the optimal and homogeneous allocations, where the cache capacity, normalized by nN, is 1%. 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 increase in proportion to the cache capacity. In other words, the remaining traffic will actually be reduced if the caches can increase proportionally with the increasing number of objects. This occurs because the number of objects *requested* increase at a sub-linear rate compared to the number of objects, allowing the caches to consolidate their capacity better. One could say that this is optimistic; as such, Fig. 7 also presents the results when maintaining a constant total cache capacity (100 entries per node on average). In this case, the remaining traffic increases approximately logarithmically with the number of content items in both allocations. Consequently, we conclude that the increasing scale of the content will, indeed, need to be matched by increasing the scale of caching resources. As caches are co-located with the routing infrastructure, this will not be trivial in terms of deployment; it certainly will not be as easy as expanding resources in a traditional Web cache.

F. Impact of cache replacement policy

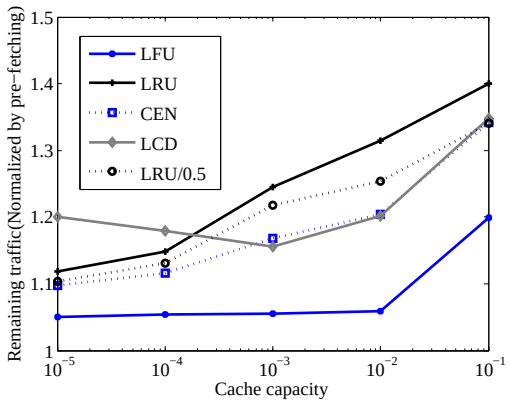
As mentioned in the previous sections, the optimal cache allocation alone cannot guarantee optimal cache performance in CCN. Instead, it must be combined with a suitable replacement strategy. As the theoretical optimal allocation algorithm in this paper is obtained using optimal object placement, it 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. Furthermore, to study the impact of allocation on the replacement policy, we also repeat the same experiment with the homogeneous allocation.

To illustrate the relative cache performance compared to the pre-fetching method (exact optimal placement), the remaining traffic on the y-axis is normalized using that achieved by the optimal placement. Besides the default cache replacement strategy (LFU), we also experiment with several other algorithms: (1) Least Recently Used (LRU) and "cache everything everywhere" [1]; (2) LRU with the centrality-based caching policy [18] (CEN); (3) LRU with LCD caching policy [35]; and (4) LRU with fixed caching probability ($p = 0.5$).

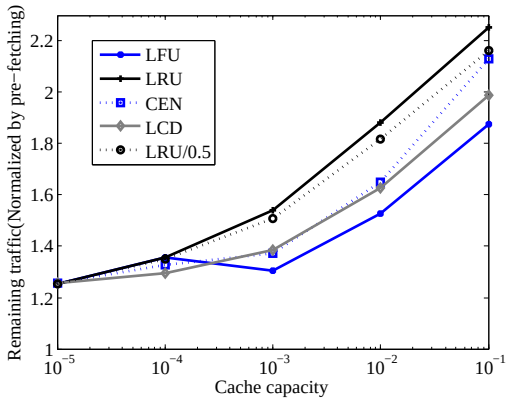
Fig. 8 shows that the caching performance of the optimal allocation is much better than the homogeneous allocation for all replacement strategies. Therefore, the benefits of heterogeneous optimal cache placement does *not* require oracle-based pre-fetching: significant gains can be made with various 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 [19]. However, with the optimal cache allocation, it does worst at these low capacity levels. 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).

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%. 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 (like 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 best fit replacement strategy according to their network environment as well as the cache allocation method in use.



(a) Optimal allocation.



(b) Homogeneous allocation.

Fig. 8. Caching performance of different replacement strategies. The cache performance of replacement strategies are normalized by optimal oracle-based pre-fetching.

V. Summary of Findings

The previous section has explored the key factors that impact optimal cache allocation. This is an important tool for any network planner who is considering a CCN deployment. 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 router design should therefore ensure that cache capacity is easily pluggable and extensible to support this philosophy.
- 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 egalitarian 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 (e.g., [34]) is better handled by pushing caches into the core (e.g., an Internet Exchange Point [36]) to better aggregate requests. For highly skewed demand, caches must be pushed to the edge; this, however, creates high levels of redundancy in which many replicas of the same content are created.
- As the number of these 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 benefits 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. Cache allocation also impacts the performance of cache replacement methods.

VI. Conclusion and Future Work

In this work, an exact optimization method has been presented to find the optimal cache allocation in CCN. Using this, we have explored the many factors that affect cache placement, and how they subsequently impact performance

(measured by traffic reduction). Our experiments have shown that the benefits of heterogeneous cache allocation are significant. Through this, we have found that many aspects may affect cache performance, including topological properties and content request patterns. Importantly, this highlights that a one-size-fits-all approach will never be optimal during CCN's deployment and, instead, network operators must make well informed decisions on where they allocate capacity.

A number of interesting avenues of future work remain in this area. Our evaluation is based on a formal model of topology and workload. In the future, we need to evaluate the performance of the different caching strategies in real-world topologies with more complex workload models. For example, we wish to include request models that consider temporal and spatial locality, as well as the server and client distribution. Although including these aspects would make the model more complex, we believe it could show even better gains from caching. Last, it is also necessary to investigate other performance metrics beyond traffic removal, e.g., delay and transit costs. We will further connect this to more real-world concerns, such as the impact that this might have on inter-AS business arrangements and incentives.

VII. Acknowledgments

This work was supported by the National Basic Research Program of China with Grant 2012CB315801, the National Natural Science Foundation of China (NSFC) with Grants 61133015 and 61272473, the National High-tech R&D Program of China with Grant 2013AA013501, and by the Strategic Priority Research Program of CAS with Grant XDA06010303. The work was also supported by the EC EINS and EPSRC IU-ATC projects.

References

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

[2] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of ccn-caching trees," in *IFIP Networking*, 2011.

[3] 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. of IEEE ICCCN*, 2012.

[4] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. of the first workshop on Information-centric networking*, 2011.

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

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

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

[8] A. Jiang and J. Bruck, "Optimal content placement for en-route web caching," in *Proc. of IEEE NCA*, 2003, pp. 9–16.

[9] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *ACM CoNEXT*, 2010.

[10] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *IEEE INFOCOM*, 2010.

[11] D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," in *Proceedings of IEEE INFOCOM NOMEN workshop*, 2012.

[12] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second workshop on Information-centric networking*, 2012.

[13] 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 *ACM SIGCOMM*, 2013.

[14] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011.

[15] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *IEEE INFOCOM*, 2010.

[16] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: hamlet-like doubts in icn," in *Proceedings of the second workshop on Information-centric networking*, 2012.

[17] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proc. of NOMEN workshop*, 2012, pp. 310–315.

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

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

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

[21] R. Shah, "Faster algorithms for k-median problem on trees with smaller heights," 2003.

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

[23] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: Ndn simulator for ns-3," *University of California, Los Angeles, Tech. Rep*, 2012.

[24] (2012) The ccnsim homepage. [Online]. Available: http://perso.telecom-paristech.fr/ drossi/index.php?n=Software.ccnSim

[25] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The origin of power laws in Internet topologies revisited," in *IEEE INFOCOM*, 2002.

[26] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *ACM SIGCOMM*, 1999.

[27] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: degree-based vs. structural," in *ACM SIGCOMM*, 2002.

[28] A. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of'small-world'networks." *Nature*, vol. 393, no. 6684, pp. 409–10, 1998.

[30] B. Quoitin, V. V. den Schrieck, P. Francois, and O. Bonaventure, "IGen: Generation of Router-level Internet Topologies through Network Design Heuristics," in *Proceedings of the 21st International Teletraffic Congress*, 2009, pp. 1 – 8.

[31] 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, 2009.

[32] 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 *Proceedings of the 2012 ACM conference on Internet measurement conference*, ser. IMC '12.

[33] N. Sastry, "How to tell head from tail in user-generated content corpora," in *Proc. International Conference on Weblogs and Social Media (ICWSM)*, 2012.

[34] H. Abrahamsson and M. Nordmark, "Program popularity and viewer behaviour in a large tv-on-demand system," in *ACM IMC*, 2012.

[35] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[36] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large european ixp," in *ACM SIGCOMM*, 2012.

View publication stats