Computer Communications 36 (2013) 758–770

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Cache "less for more" in information-centric networks (extended version) ☆,☆☆

Wei Koong Chai *, Diliang He, Ioannis Psaras, George Pavlou

Department of Electronic and Electrical Engineering, University College London, WC1E 6BT, Gower Street, London, UK

ARTICLE INFO

Article history:
Available online 29 January 2013

Keywords:
Information-centric networking
Caching
Betweenness centrality

ABSTRACT

Ubiquitous in-network caching is one of the key aspects of information-centric networking (ICN) which has received widespread research interest in recent years. In one of the key relevant proposals known as Content-Centric Networking (CCN), the premise is that leveraging in-network caching to store content in every node along the delivery path can enhance content delivery. We question such an indiscriminate universal caching strategy and investigate whether caching less can actually achieve more. More specifically, we study the problem of en route caching and investigate if caching in only a subset of nodes along the delivery path can achieve better performance in terms of cache and server hit rates. We first study the behavior of CCN's ubiquitous caching and observe that even naïve random caching at a single intermediate node along the delivery path can achieve similar and, under certain conditions, even better caching gain. Motivated by this, we propose a centrality-based caching algorithm by exploiting the concept of (ego network) betweenness centrality to improve the caching gain and eliminate the uncertainty in the performance of the simplistic random caching strategy. Our results suggest that our solution can consistently achieve better gain across both synthetic and real network topologies that have different structural properties. We further find that the effectiveness of our solution is correlated to the precise structure of the network topology whereby the scheme is effective in topologies that exhibit power law betweenness distribution (as in Internet AS and WWW networks).

1. Introduction

Information-centric networking (ICN) has recently attracted significant attention, with various research initiatives (*e.g.*, DONA [1], CCN/NDN [2], PSIRP/PURSUIT [3,4] and COMET [5–7]) targetting this emerging research area. The main reasoning for advocating the departure from the current host-to-host communications paradigm to an information/content-centric one is that the Internet is currently mostly being used for content access and delivery, with a high volume of digital content (*e.g.*, movies, short videos, photos *etc.*) delivered to users who are only interested in the actual content itself rather than the hosting server location. While the Internet was designed for and still focuses on host-to-host communication, ICN shifts the emphasis to content objects that can be cached and accessed from anywhere within the network rather than from the end hosts only.

In ICN, content names are decoupled from host addresses, effectively separating the role of identifier and locator in distinct contrast to current IP addresses which are serving both purposes. Naming content directly enables the exploitation of in-network caching in order to improve delivery of popular content. Each content object can now be uniquely identified and authenticated without being associated to a specific host. This enables application-independent caching of content pieces that can be re-used by other end users requesting the same content. In fact, one of the salient ICN features is in-network caching, with potentially every network element (*i.e.*, router) caching all content fragments[1] that traverse it; in this context, if a matching request is received while a fragment is still in its cache store, it will be forwarded to the requester from that network element, avoiding going all the way to the hosting server. Out of the current ICN approaches, content-centric networking (CCN) [2] advocates such indiscriminate content caching.

We argue that such an indiscriminate universal caching strategy is unnecessarily costly and sub-optimal and attempt to study alternative in-network caching strategies for enhancing the overall content delivery performance so that network bandwidth consumption is further reduced, server load is further alleviated and delays

☆ An earlier abbreviated version of this paper was presented at the IFIP Networking 2012, Prague, Czech Republic, 23 May 2012 [12]. It was awarded the Best Paper Award.

☆☆ The research leading to these results has received funding from the EU FP7 COMET project – Content Mediator Architecture for Content-Aware Networks, under Grant Agreement 248784.

* Corresponding author. Tel.: +44 (0)20 7679 5722.
E-mail address: w.chai@ucl.ac.uk (W.K. Chai).

[1] In our study, the basic unit of a content can be a packet, a chunk or the entire object itself.

0140-3664/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2013.01.007

experienced by end users are further reduced. We address the central question of whether caching only at a specific sub-set of nodes en route the delivery path can achieve better gain. If yes, which are these nodes that maximize caching gain and how can we identify them?

In essence, if not adopting a universal caching strategy, ICN in-network caching is an en route caching problem which we address in this work. Content can only be cached along the routing paths from content servers to requesting users without redirection, *i.e.*, content is not rerouted to any nodes not directly involved in the content delivery path. Besides the option of ubiquitous indiscriminate caching as advocated by CCN, there are two main schools of thought. The first advocates caching content at network edges (*e.g.*, in the form of proxy caches) [8] while the second, in contrast, presents evidence that placing caches in the core backbone can obtain better caching gain [9]. While it was shown in [10] that, in its general formulation, this class of cache location problem is intractable, we have shown in our previous work in [11] that the critical point is to cache content closer to the users regardless of their relative locations in the topology. Thus, we need to solve the problem of finding caching locations along the content delivery paths that exhibit the highest probability of getting cache hits while at the same time diffusing content quickly enough to locations where the content will be popular in order to avoid flash crowd situations.

Our contributions in this study, which is a significant extension of our initial work in [12], is fourfold. First, we contribute to the understanding of ubiquitous caching in networked systems by providing insights into its behavior for specific topology types. Second, we demonstrate that selective instead of ubiquitous caching can achieve higher gain even when using simplistic random selection schemes. Third, we propose a centrality-driven caching scheme by exploiting the concept of (ego network) betweenness derived from the area of complex/social network analysis, where only selected nodes in the content delivery path cache the content. The rationale behind such a selective caching strategy is that some nodes have higher probability of getting a cache hit in comparison to others and by strategically caching the content at "better" nodes, we can decrease the cache eviction rate and, therefore, increase the overall cache hit rate. Fourth, we characterize our centrality-driven scheme and find the graph invariant that determines the effectiveness of the scheme. We found that the scheme is sensitive to the betweenness distribution of the vertices but not directly to the degree distribution.

In the next sections, we first review previous work on in-network caching, starting from pre-ICN work until the most recent research efforts. We then define the system of interest and lay-out our arguments and rationale with a motivating example illustrating that caching *less* can achieve *more*. We then explain the design and features of our centrality-based caching scheme that can consistently outperform ubiquitous caching. We carry out a systematic simulation study that explores the parameter space of the caching systems, diverging from existing work in networked caches which mostly considers topologies with highly regular structure (*e.g.*, string and tree topologies [11,13,14]); in the latter, the content source(s) are usually located at the root of the topology forcing a sense of direction on content flows for tractable modeling and approximation. We present results for both regular and non-regular topologies, including *k-ary* trees and scale-free topologies whose properties imitate closely the real Internet topology. Besides synthetic topologies, we further verify the consistency of our findings for a large-scale real Internet AS topology. In Section 6, we delve deeper into the characteristics of our proposed scheme and relate its effectiveness to the specific topology structure. We found that the caching gain is correlated with the topology structure and a power law betweenness distribution ensures a better performance of our solution compared against ubiquitous caching. We finally discuss some simple variants of our approach and its practical implications in the real world.

2. Related work

In the networking area, caching has been studied in different forms. Initial studies focused on the performance of different cache replacement policies in standalone caches [15,16]. This isolates the effect of connected caching nodes (*i.e.*, a network of caches). Caching has also been studied in the context of content distribution networks (CDNs) and in the World-Wide Web (web caching), in both cases in a network overlay fashion with some forms of collaborative (*e.g.*, cooperative / selfish caching through game theory [17,18]) or structured (*e.g.*, hierarchical caching [19,20]) approaches being considered.

In ICN, caching takes place within the network (*i.e.*, routers are equipped with cache stores), requiring line-speed operation. In fact, the idea of identifying suitable caches within the network has been investigated prior to the emergence of ICN [21–23]. But in ICN, complex algorithms involving multiple collaborating entities that require extensive information exchanges are simply not feasible at line-speed.

One of the key ICN proposals, content-centric networking (CCN) [2], defines its in-network caching strategy as follows:

- A router caches every content chunk that traverses it with the assumption that routers are equipped with (large) cache stores.
- A least recently used (LRU) cache eviction policy is used.

This ubiquitous caching strategy ensures a quick diffusion of content copies throughout the network. Hereafter, we refer to this scheme as *CCN* and treat it as the benchmark for performance comparison. In addition to that, LRU is used as the cache eviction policy in all the different caching schemes described in this paper.[2]

There are several recent studies on this *CCN* caching strategy. In our previous work [11], we model it with a continuous time Markov-chain and assess the proportion of time a given piece of content is cached. [13] derives a closed form expression for average content delivery time under the same caching scheme. This work focuses on content popularity and investigates cache partitioning in order to maximize the gain from in-network caching. On the other hand, [24] proposes an algorithm to approximate the behavior of a more general multi-cache setup under arbitrary topologies.

In [25], the authors perform a comprehensive simulation study of *CCN* under various topologies, content popularity distributions, catalog sizes and replacement policies. They conclude that content popularity is by far the most important factor that affects the cache hit performance. Authors in [26] investigate the impact of traffic mix on the caching performance of a two-level cache hierarchy. Web, file sharing, user generated content and video on demand (VoD) have been identified as the four main types of content. They conclude that caching performance increases if VoD content is cached towards the edge of the network (*i.e.*, at the leaf cache of the two-level cache hierarchy) in order to leave core with large caches for other types of content.

Contradictory findings also appear in the literature. In [27], the authors use topological information to size router caches proportional to different centrality metrics and find that the gain achievable with heterogeneous cache sizes is limited. Note that while we also exploit the concept of centrality in this paper, it is for entirely different purpose, *i.e.*, we base the actual caching decision on the node centrality while [27] uses it to determine the size of the

[2] For readability, we omit "LRU" when labeling the caching schemes discussed.

cache. In contrast, [28] found that caches of up to 10,000 packets have significant benefits with respect to both cache hit and path length under a trace-driven simulation analysis based on BitTorrent traces emulating request patterns in real-world settings.

Besides studies on *CCN*, others have already started investigating algorithms that make more efficient use of caching resources. In [29], the authors proposed to cache chunks of the most popular content only. Popularity here is based on request count. As the request count increases, the algorithm increases (exponentially) the number of chunks to cache from this content. Unlike our proposed approach here, this algorithm ignores completely the topological features of the network.

In our most recent work [30], we design a distributed caching algorithm, which tracks the number of hops that a content travels from source to destination. Based on this, we approximate the caching capability of the path and decide probabilistically whether to cache or not incoming content; therefore, decisions are based on the router's distance from the content's destination. The algorithm multiplexes flows in the available caching space based on their path lengths to maximize caching gain.

Finally, [31] argues on the necessity of advertising cached content in the control plane. In particular, they argue that making distributed caching decisions in an uncooperative fashion will fail to exploit the potential advantages in-network caching can bring. However, the scalability and performance of their approach operating at line-speed is a serious question while the extra benefit of the cooperation is yet to be assessed.

3. Model description

As a foundation, we first assume that the network has an ICN publish/subscribe framework in place (*e.g.*, [1–7]). Specifically, we assume that a content request and resolution mechanism is already in place. As pointed out in [32], all the different ICN proposals in the literature have invariably such common functions (although typically using different primitives). Let $G = (V, E)$ be an undirected network with $V = v_1, \ldots, v_N$ nodes and $E = e_1, \ldots, e_M$ links. We denote $F = f_1, \ldots, f_R$ the content population in the system and $S = s_1, \ldots, s_P$ the set of content servers, each associated to a $v \in V$. The content population is randomly hosted in S and we assume that each content object is hosted permanently in only one server.

Content requests are assumed to arrive in the network exogenously and the content request arrival process for content unit $r, 1 \leqslant r \leqslant R$, follows the Poisson process with mean rate, $\lambda = \sum_{r=1}^{R} \lambda_r$, whereby λ_r is the rate of exogenous content request for f_r. A *cache hit* is recorded for a request finding a matching content along the content delivery path. Otherwise, a *cache miss* is recorded. In the event of a cache miss, the content request traverses the full content delivery path to the content server. Following the convention in the literature, we assume that content units are of the same size and each cache slot in a cache store can accommodate one content unit at any given time. When a cache store is full, the least recently used content will be discarded in the event of an arrival of a new uncached content.

The objectives of this study are: (1) to examine the caching performance of such a system under different caching schemes, (2) to gain insights into the behavior of ubiquitous caching and (3) to develop and understand more sophisticated caching algorithms for achieving better gain.

4. Centrality-based in-network caching scheme

4.1. Design rationale

The ubiquitous indiscriminate caching scheme has already raised doubts (*e.g.*, [32]). In the general cache-related literature, some authors have already questioned this aggressive *cache-everything-everywhere* strategy [19,20,33]. The basic reasoning is that since the caching capacity is usually much smaller than the overall population of the items to be cached, it has the property of high cache replacement error. We illustrate this property of ubiquitous caching with a motivating example. We define a naïve random caching strategy, *Rdm*, which simply caches randomly at only one intermediate node along the delivery path per request, using LRU cache eviction policy. We compare the two caching schemes in a 7-node string topology where $P = 1, s_1$, is located at v_1 (root) while content requests originate exogenously from other nodes. The detail simulation setup is described in Section 5.1. We observe, in Fig. 1, that even random caching at just a single node along the content delivery path can reduce both the number of hops required to hit the content and the server hits in comparison to ubiquitous caching (*CCN*). The reduction ratios are defined in Section 5.1.

Based on the above observations, we realize that caching indiscriminately does not necessarily guarantee the highest cache hit rate. Intuitively, this may be caused by the high cache replacement rate on non-selective caching schemes such as *CCN*. By caching indiscriminately, a content in a cache is more likely to be replaced before it gets a hit. On the other hand, this result cannot be used as conclusive evidence that caching less is better since the string topology constrains to quite a large extent the diversity of the content delivery paths (*i.e.*, all delivery paths are fully or partially overlapping), a fact that indirectly increases the probability of a cache

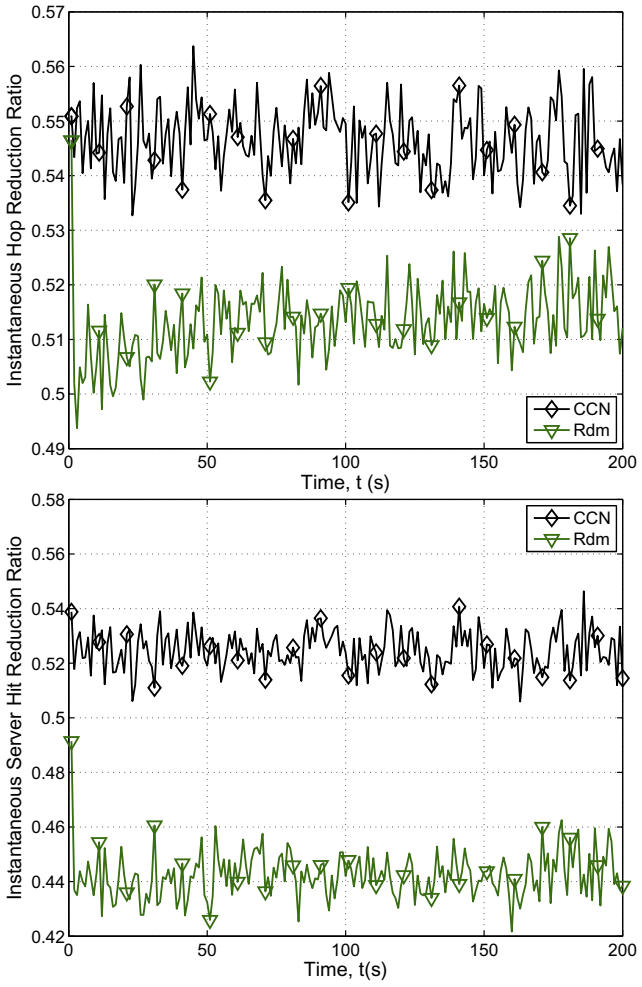

Fig. 1. Simple random caching outperforming ubiquitous caching in the number of hops to hit the content (top) and reduced server hits (bottom).

hit. In Section 5, we study and discuss such phenomenon in detail with the support of empirical results.

Following this line of argument, we design a novel caching scheme with the following features in mind. First, the caching scheme is able to cache at locations with high probability of getting cache hits. For example, core nodes within a network may have higher chances of getting a cache hit since they are connected to many others while edge nodes are more sparsely connected. Second, the caching scheme is able to spread content towards users rapidly by exploiting topological features of the network. In [11], we already showed that caching content closer to network edges may yield higher gains. Since content exhibits localized popularity [34], it is important for the caching scheme to be able to spread content to the network region where it will be highly popular. Third, the solution should be lightweight since in-network caching operates at line-speed. Caching decisions requiring complex computations or interactions with other entities would render the scheme too slow.

4.2. Betweenness centrality caching scheme

Our solution is based on the concept of betweenness centrality [35] which measures the number of times a specific node lies on the content delivery paths between all pairs of nodes in a network topology. There are various definitions of centrality in the literature (*e.g.*, degree, closeness, eigenvector centrality *etc.*). However, the betweenness centrality seems to find more natural use in communication networks. In [36,37], which investigate load distribution in weighted complex networks, a quantity called *load* is defined to measure the burden of vertices in the shortest path-based transport processes. It is found that while the load is a dynamic quantity, it is closely related to the static quantity of betweenness centrality and thus, betweenness centrality appears to be a good measure of node importance [39].

In the context of in-network caching, the basic intuition is that if a node lies along a high number of content delivery paths (*i.e.*, having high betweenness centrality), then it is more likely to get a cache hit. By caching only at those more "important" nodes along the delivery paths, we reduce the cache replacement rate while still caching content where a cache hit is most likely to happen.

Let's consider the topology in Fig. 2. At time $t = 0$, all cache stores are empty and client A requests a content from s_1. The content is being routed via $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ from s_1 to client A. With *CCN*, all four nodes will retain a copy of the content while under *Rdm*, only one of them will cache the content. Let's assume now that client B requests the same content. For *CCN*, the request is satisfied by v_3 but the cached copies at v_1, v_2 and v_4 are redundant. On the other hand, under *Rdm*, there is $\frac{1}{4}$ chance to get a cache miss (*i.e.*, content cached at v_4) and $\frac{1}{2}$ chance that the hop count reduction is worse than *CCN* (*i.e.*, the copy is cached at either v_1 or v_2). However, with a bird's eye view, it is clear that caching the content only at v_3 is sufficient to achieve the best gain without caching redundancy at other nodes. This can be verified by using the betweenness centrality, whereby v_3 has the highest centrality value with most content delivery paths passes through it (*i.e.*, 9 paths).

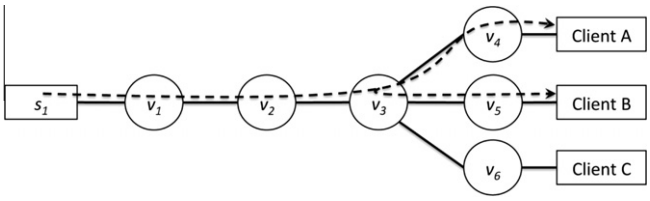

Fig. 2. A topology with optimal caching location at v_3.

We now present our algorithm which we will call *Betw* hereafter. We assume that the betweenness centrality of each node is pre-computed offline (*e.g.*, by the central network management system) as follows.

$$\text{betweenness centrality, } C_B(v) = \sum_{i \neq v \neq j \in V} \frac{\sigma_{i,j}(v)}{\sigma_{i,j}} \quad (1)$$

where $\sigma_{i,j}$ is the number of content delivery paths from i to j and $\sigma_{i,j}(v)$ is the number of content delivery paths from i to j that pass through node v. Without loss of generality, we use the shortest path as the content delivery path in this paper. More sophisticated content delivery path computation (*e.g.*, [38]) can also be used.

Betw operates at per request level whereby the selected caching node may differ from one delivery path to another. Hence, there is no *fixed* pre-configured caching node in the network (*e.g.*, solutions to *k*-median problems). Specifically, when a content client initiates a content delivery, the request message (*e.g.*, `Find` in [1], `Interest` in [2], `Consume` in [7]) records the highest centrality value among all the intermediate nodes it traverses. This value is copied onto the content messages during the data transmission at the server. The same applies to the router where the request message found the content before reaching the server (*i.e.*, a cache hit). On the way to the requesting user, each router matches its own C_B against the attached one and the content is cached only if the two values match. If more nodes have the same highest centrality value, all of them will cache the content. Note that our solution is highly lightweight as each node makes its caching decision independently, solely based on its own C_B, neither requiring information exchange with other nodes nor inference of server location or of traffic patterns, as it is the case with collaborative or cooperative caching schemes. In this case, the C_B value is pre-computed offline and configured to every router by the network management system. The pseudo-code for forwarding both the request and the actual content is shown on Table 1.

A desired property of this scheme is that a content will be *pulled* to the region where it is popular (*i.e.*, frequently requested). Although for a fixed network topology, the betweenness value of each node remains static, the caching of content does not necessarily concentrate at several nodes that have very high centrality values (*e.g.*, in core routers). This is because our scheme ensures the spreading of content towards the origin of content requests. For example, a content delivery path routes content along $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ with $C_B(v_1) > C_B(v_2) > C_B(v_3) > C_B(v_4)$. The first request from v_4 will have its content cached at v_1 since v_1 has the highest C_B along the path. Assuming that the content is not evicted from v_1, the second request from v_4 will have the content cached at v_2 as it will hit the requested content at v_1 and the highest C_B recorded in the request message will be $C_B(v_2)$ (*i.e.*, the highest C_B from the remaining section of the delivery path). Following a similar procedure, the third request will be cached at v_3 and thus, if the content is repetitiously requested from one edge, then a copy of this content will be cached closer to the clients. Thus, content copies always move towards the last content consumer and if a content is popular at a specific topology region, it will eventually be cached in that region.

4.3. Distributed centrality computation via approximation

We now sketch a distributed implementation for *Betw* where the full network topology may not be readily available because of an infrastructure-less network with relatively dynamic topology (*e.g.*, self-organizing, ad hoc and mobile networks). Since in this case it is not practical for dynamic nodes to efficiently obtain the knowledge of delivery paths between all pairs of nodes in the network, we envision that the nodes themselves can compute an

Table 1
Pseudo-code for both content request and data.

```
Content request
1.Initialize (C_B=0)
2.foreach (v_n from i to j)
3.   if data in cache
4.     then send (data)
5.   else
6.     Get C_B(v_n)
7.     if C_B(v_n) > C_B
8.       then C_B = C_B(v_n)
9.     forward request to the next hop towards j
```

```
Content data
1.Record C_B from corresponding content request
2.foreach (v_n from j to i)
3.   Get C_B(v_n)
4.   if C_B(v_n) == C_B
5.     then cache (data)
6.forward data packet to the next hop towards i
```

approximation of their C_B. This approximation is based on the ego network betweenness concept [40]. The ego network consists of a node together with all of its immediate neighbors and all the links among those nodes. The idea is for each node, v to compute its $C_B(v)$ based on its ego network rather than the entire network topology. From [40], if A is the $N \times N$ symmetric adjacency matrix of G, with $A_{i,j} = 1$ if there exists a link between i and j and 0 otherwise, then $A^2[\mathbf{1} - A]_{i,j}$, where $\mathbf{1}$ is a matrix of 1's, gives the number of 2-hop paths joining i and j. The ego betweenness is then the sum of the reciprocal of the entries.

From an implementation point of view, the construction of the ego network for each node can be done by simply requiring each node to broadcast the list of its one-hop neighbors with message Time-To-Live = 1 when it first joins the network and whenever there are changes to its one-hop neighbor set. The overhead is thus limited as the message propagation is limited to one hop only. If d_i is the number of neighbors node i has, then the extra messages to construct the ego network for all the nodes in a topology is equivalent to $\sum_{i=1}^{N} d_i$. In the case of a dynamic topology (*e.g.*, ad hoc networks), each change will incur an additional overhead, amounting to two times the "moving" node's new degree (since each neighbor has to respond to this newly arrived node).

The ego network can then be built by adding links that connect to itself or its own neighbors based on the received neighbor lists and ignoring the entries to nodes not directly connected to itself. The ego network betweenness is simply the $\sum \sigma_{i,j}(v)/\sigma_{i,j}$ of v's ego network. The rest of the caching operations remain unchanged, *i.e.*, as described in the previous section.

Although the ego network betweenness only reflects the importance of a node within its ego network, it has been found that it is highly correlated with its betweenness centrality counterpart in real-world Internet service provider (ISP) topologies [41]. Coupled with its low computation complexity (reduced from $O(NM)$[3] to $O(d_{max}^2)$ where d_{max}^2 is the highest node degree in the network), it presents itself as a good alternative for large / dynamic networks. This caching algorithm using ego network betweenness centrality along with the LRU cache eviction policy is referred to as *EgoBetw* hereafter. Referring back to Fig. 2, the outcome of *Betw* and *EgoBetw* is the same since v_3 remains the node having the highest centrality value.

5. Performance evaluation

5.1. Performance metrics and evaluation methodology

We use a custom-built simulator for the study of the dynamics of content caching in order to evaluate our proposal. All nodes in the simulator are cache-enabled and we perform the experiments based on the specifications described above for the different caching schemes.

Caching in networks aims to: (1) lower the content delivery latency whereby a cached content near the client can be fetched faster than from the server, (2) reduce traffic and congestion since content traverses fewer links when there is a cache hit and (3) alleviate server load as every cache hit means serving one less request. We use the *hop reduction ratio*, β as the metric to assess the effect of the different caching schemes on (1) and (2) above while we use the *server hit reduction ratio*, γ on (3).

$$\text{Hop Reduction Ratio, } \beta(t) = \frac{\sum_{r=1}^{R} h_r(t)}{\sum_{r=1}^{R} H_r(t)} \tag{2}$$

where $H_r(t)$ is the path length (in hop count) from client(s) to server(s) requesting f_r from time $t-1$ to t and $h_r(t)$ is the hop count from the content client to the first node where a cache hit occurs for f_r from $t-1$ to t. If no matching cache is found along the path to the server, then $h_r = H_r$. In other words, the hop reduction ratio counts the percentage of the path length to the server used to hit the content given caching in intermediate nodes. In a non-caching system, $\beta = 1.0$.

$$\text{Server Hit Reduction Ratio, } \gamma(t) = \frac{\sum_{r=1}^{R} w_r(t)}{\sum_{r=1}^{R} W_r(t)} \tag{3}$$

where $W_r(t)$ is the number of requests for f_r from $t-1$ to t and $w_r(t)$ is the number of server hits for f_r from time $t-1$ to t. Note that high hop reduction does not directly translate to high server hit reduction.

We seek to draw insights from the inspection of network topologies with very different structural properties – (1) *k-ary* trees which have almost strict regular structure (*i.e.*, all nodes besides the root and leaves have the same $k+1$ degree) and (2) scale-free topologies following the Barabasi–Albert (B-A) power law model [42] which accounts for the preferential attachment property of the Internet topology and results in graphs with highly skewed degree distribution. It is interesting to note that the betweenness distribution of B-A graphs also follows the power law model [39].

Content requests for different content are generated based on Zipf-distribution with $\sum_{r=1}^{R}(\frac{C}{r^{\alpha}}) = 1$ where the probability for a request for the rth popular content is C/r^{α} with α being the popularity factor. We use $\alpha = 1.0$ and requests originate randomly from all nodes.[4] Each simulation run begins with all cache stores being empty (*i.e.*, cold start). The content population is randomly distributed in the network with each content object being hosted persistently in one server. Unless otherwise specified, the simulations are run with the following parameters: total simulation time = 200 s, λ = 5000 request/s, R = 1000 and uniform cache store size = 100 content.

5.2. k-ary tree topologies

5.2.1. Instantaneous behavior

A *k-ary* tree is defined via two parameters, namely k, the spread factor, denoting the number of children each node has and D is the

[3] Based on the best known betweenness computation algorithm in U. Brandes, "A faster algorithm for betweenness centrality", Journal of Mathematical Sociology 25(2):163–177.

[4] From our results, we note that the order of performance amongst the caching schemes remains unchanged for $0.6 \leqslant \alpha \leqslant 1.5$. So, the results presented here are valid for these values of α.

depth of the tree from root. We show in Fig. 3 the instantaneous behavior of the different caching schemes for both β and γ in a 5-level binary tree ($k = 2, D = 4$). All caching schemes reach a stationary performance after a few seconds. We point out that since all simulations go through a *warm-up* phase, *CCN* always reaches the stable performance level first. This is due to its "*always cache*" policy.

We observe that both *Betw* and *Rdm* perform better than *CCN* for both metrics. Tracking the evolution of the cache stores over time revealed that this is due to the high cache replacement rate in *CCN*. Replacing cached content rapidly causes content often being evicted before the next matching request is received. The effect is magnified considering that the whole chain of caches on the delivery path is affected. This is the fundamental basis on why the counter-intuitive *"less for more"* caching scheme proposed here can be true. We further observe that the argument that caching selectively may increase cache miss is untrue in *k-ary* trees. We do find that there are more cache misses if the caching node is randomly selected rather than caching at nodes with high betweenness. Finally, an interesting observation is that instead of approximating the performance of the *Betw* scheme as it was meant to be, *EgoBetw* actually performs at the same level as *CCN*. This is due to the regularity of the topology whereby nodes between the root and the leaves have the same ego network and thus, have the same C_B. Since the algorithm specifies that all nodes with equal highest C_B along the delivery path should cache, in this case *EgoBetw* is simply reduced to a similar behavior with *CCN*.

5.2.2. Effect of topology features on performance

In *k-ary* trees, *D* affects the expected path lengths and *k* impacts the path diversity. We now study the validity of the previous observations in different configurations of *k-ary* trees by obtaining the β at 95% confidence interval for a range of depths and spread factors. Our results in Fig. 4 suggest that the caching schemes exhibit consistent behavior for different *k-ary* trees.

We find that while the performance distance between *CCN* and *Betw* remains approximately constant, *Rdm* does not exhibit such consistency. In general, *Rdm* always has the highest variance due to the randomness implicit to the algorithm. *Rdm* performs increasingly better in terms of hops saved when *D* is increased and *k* is decreased. This is due to the fact that each node has equal probability to cache content and in effect, distributes cache replacement operation uniformly across different nodes. In turn, this results in content being cached longer when compared to *CCN*. It increases the cache hit probability especially in topologies with very low number of content delivery paths. This, however, is counter-balanced by the increased number of branches in the topology, whereby a greater number of cache misses will occur. Our *Betw* scheme does not suffer from such a drawback since the caching node always has the highest probability of getting a cache hit and thus maintains stable cache hit (reducing server hits) and network resource gain (reducing the content delivery hop count).

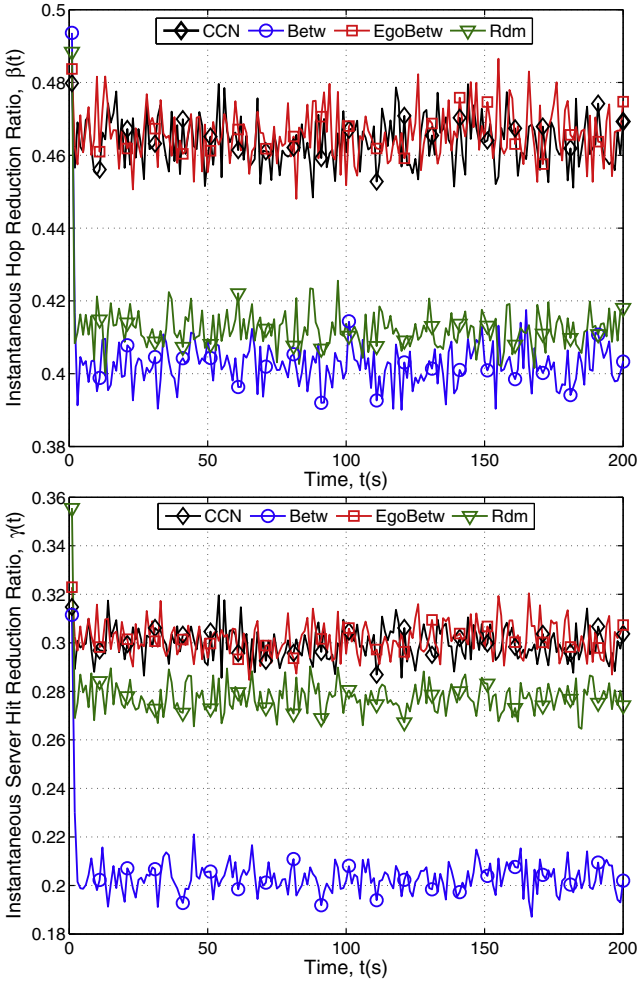

Fig. 3. Instantaneous behavior of the caching schemes for a binary tree; (top) β, (bottom) γ.

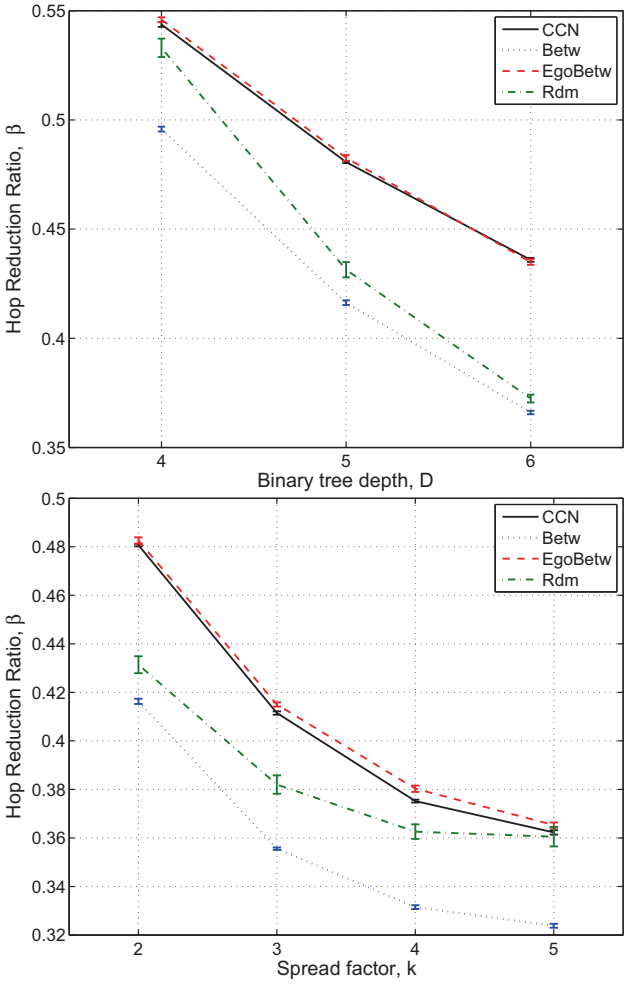

Fig. 4. *Betw* consistently outperforms the rest over different *D* (top) and *k* (bottom).

5.3. Scale-free topologies

5.3.1. Instantaneous behavior

Although regular graphs lend themselves to tractability in modeling, real-world Internet topologies are not regular but follow a power law degree distribution [42]. As such, we consider scale-free topologies following the construction method described in [42] (referred to as B-A graphs hereafter). We show in Fig. 5 the performance of the different caching schemes in a B-A graph with $N = 100$ over time. First and foremost, we see that the performance of both our centrality-based caching schemes (*Betw* and *EgoBetw*) perform better than *CCN* for both metrics and *EgoBetw* now approximates closely *Betw*. This is because, without the regular structure, the ego networks of the nodes within the B-A graphs reflect correctly their actual betweenness. This result, thus, suggests that the more scalable and distributed *EgoBetw* algorithm can be used for irregular graphs.

Second, we observe that *Rdm* no longer outperforms *CCN*. In fact, it performs at the same level as *CCN* with respect to hop reduction and due to the highly skewed degree distribution in the topology, it fails to alleviate load from the server (*i.e.*, it has the highest number of cache misses).

5.3.2. Effect of topology features on performance

Unlike *k-ary* trees which are fully described via the tuple (k, D), each generation of a B-A graph with the same parameters results in a different topology since the links are created based on a probability proportional to the attractiveness of existing nodes (*i.e.*, preferential attachment). We evaluate the caching schemes over 50 B-A graphs with $N = 100$ and mean degree = 2. From Fig. 6, both centrality-based caching schemes always perform better than the rest. The mean β achieved for *CCN*, *Betw*, *EgoBetw* and *Rdm* are 0.47581, 0.44583, 0.44845 and 0.47891 respectively. The variances obtained are 2.86357×10^{-4} (*CCN*), 4.91978×10^{-4} (*Betw*), 4.83752×10^{-4} (*EgoBetw*) and 8.83494×10^{-4} (*Rdm*). As expected, *Rdm* has the highest variance. *Rdm* is worse than *CCN* in many cases, even with the topology having the same properties. This is due to the skewed node degree distribution of the graph that increases the probability of the scheme caching at nodes having low cache hit probability. Fig. 7 shows how ego network betweenness approximates betweenness in a B-A graph.

From Fig. 8 (top), we observe again that centrality-based caching schemes provide the best hop reduction ratio while *Rdm* exhibits inconsistent gain across B-A graphs with different sizes. We observe that as the size of the topology increases, *Rdm* gradually performs worse than *CCN*. The power-law distribution of betweenness in B-A graphs plays a vital role in this phenomenon as it results in high number of nodes having low probability of getting a cache hit. Since *Rdm* does not differentiate the centrality of the

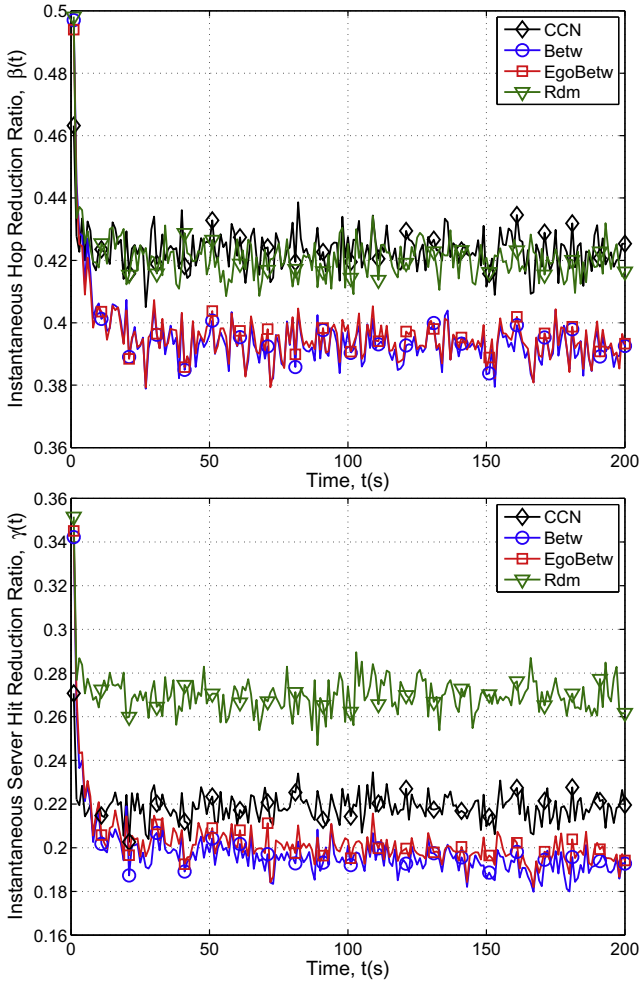

Fig. 5. Instantaneous behavior of the caching schemes for a B-A graph; (top) β, (bottom) γ.

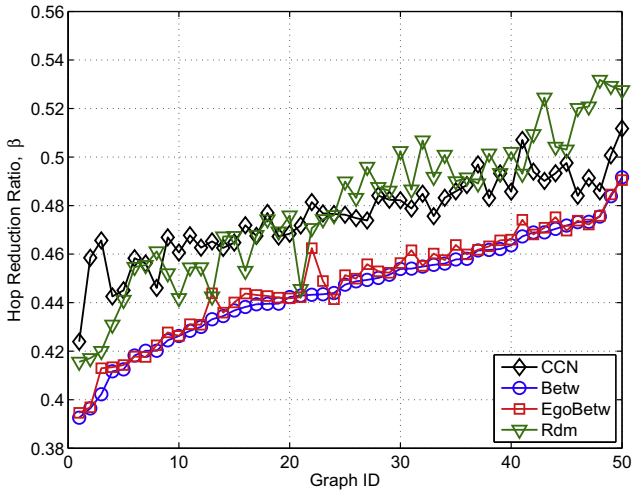

Fig. 6. Performance with different B-A graphs ($N = 100$).

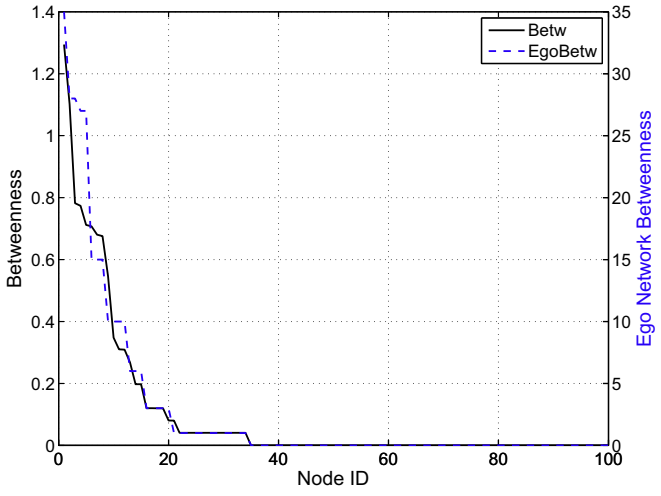

Fig. 7. A sample ego network betweenness and betweenness values of the nodes in a B-A graph.

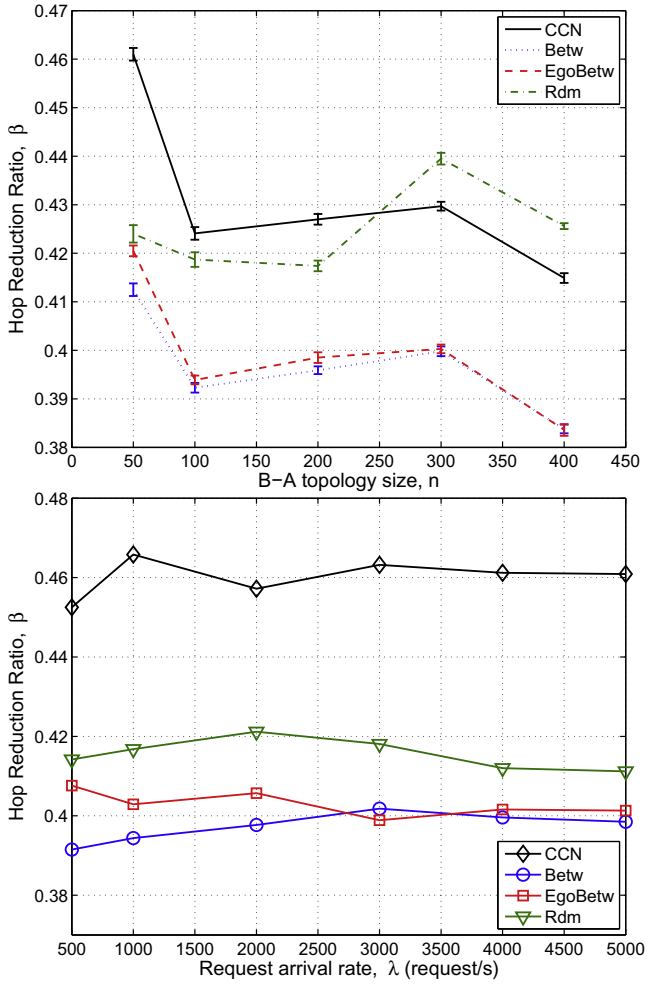

Fig. 8. Hop reduction ratio for different B-A graph sizes (top) and request rates, λ (bottom).

nodes, there is higher probability of *Rdm* caching at these *"unimportant"* nodes. Note that this observation is untrue for *k-ary* trees (the case when *D* is increased) due to the high number of overlapping shortest paths (an obvious example being the string topology).

From Fig. 8 (bottom), we see that different request intensities do not affect the order of performance amongst the caching schemes. This is due to the fact that all caching schemes converge to a stable performance level (cf., Figs. 1, 3 and 5).

In Table 2, we provide representative results of the different caching schemes across the different topologies in terms of number of hops and server hits saved. It is clear that *Betw* reliably achieves better gains (both in terms of hop and server hit reduction) in comparison to *CCN*. For instance, it reduces server hits over 30% and hop count over 17% in comparison to *CCN* in the string topology.

5.4. Real AS-level topologies

To further verify our findings, we proceed to assess the caching performance of the different caching schemes in a real-world Internet topology. We focus on a large domain-level topology, extracting a sub-topology from the CAIDA dataset [43]. The topology is rooted at a tier-1 ISP (AS7018) and contains 6,804 domains and 10,205 links. We do not aggregate stub domains while sibling domains/ links are not considered. In a similar manner to the previous simulation setup, all content servers and clients are randomly distributed across the topology. Fig. 9 shows both the hop reduction and server hit reduction ratios achieved in this setup.

The results show that the different caching schemes behave in a similar fashion to the B-A graphs but not to *k-ary* trees, reinforcing

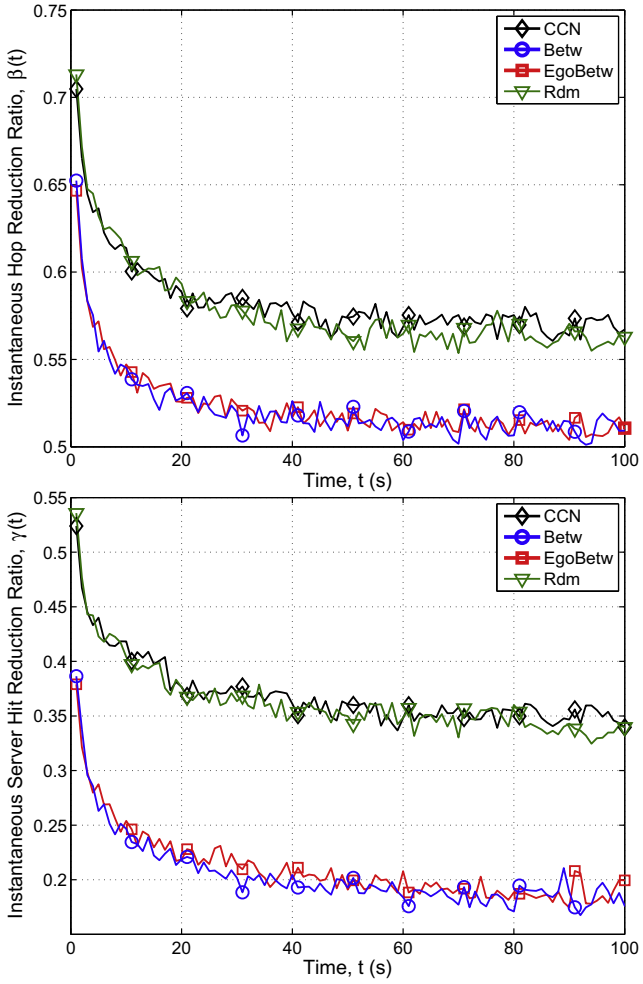

Fig. 9. Instantaneous behavior of the caching schemes in a large-scale real Internet topology; (top) β, (bottom) γ.

Table 2
Sample performance achieved after 200s in different types of topology.

Caching scheme	String ($D = 10$, $k = 1$)		*k-ary* Tree ($D = 4$, $k = 2$)		B-A ($N = 100$)	
	$\sum h$	$\sum w$	$\sum h$	$\sum w$	$\sum h$	$\sum w$
CCN	2,683,945	498,603	2,684,325	299,657	2,137,015	211,852
Betw	2,211,248	337,362	2,331,061	203,673	2,045,852	204,479
EgoBetw	2,680,614	497,146	2,698,153	301,797	2,074,089	207,628
Rdm	2,206,002	377,289	2,386,569	277,575	2,195,303	291,560

Table 3
The mean and variance of the performance achieved after the initial transient phase in different types of topology.

Metric	Caching scheme	*k-ary* Tree ($D = 4$, $k = 2$)		B-A ($N = 100$)		Real Internet ($N = 6804$)	
		Mean	Variance	Mean	Variance	Mean	Variance
β	*CCN*	0.46346	4.24717×10^{-5}	0.42315	3.01174×10^{-5}	0.56849	3.31567×10^{-5}
	Betw	0.40216	2.78626×10^{-5}	0.39235	2.58357×10^{-5}	0.51832	3.55136×10^{-5}
	EgoBetw	0.46580	4.64435×10^{-5}	0.39390	2.5936×10^{-5}	0.51605	2.81248×10^{-5}
	Rdm	0.41187	2.044×10^{-5}	0.41981	2.59586×10^{-5}	0.55843	1.90826×10^{-5}
γ	*CCN*	0.29924	4.36482×10^{-5}	0.21869	3.2002×10^{-5}	0.35137	4.67507×10^{-5}
	Betw	0.20298	2.89143×10^{-5}	0.19390	3.24316×10^{-5}	0.19005	1.20289×10^{-4}
	EgoBetw	0.30121	4.64179×10^{-5}	0.19877	3.02138×10^{-5}	0.18697	1.14169×10^{-4}
	Rdm	0.27729	3.00353×10^{-5}	0.26913	5.30707×10^{-5}	0.33394	1.28169×10^{-4}

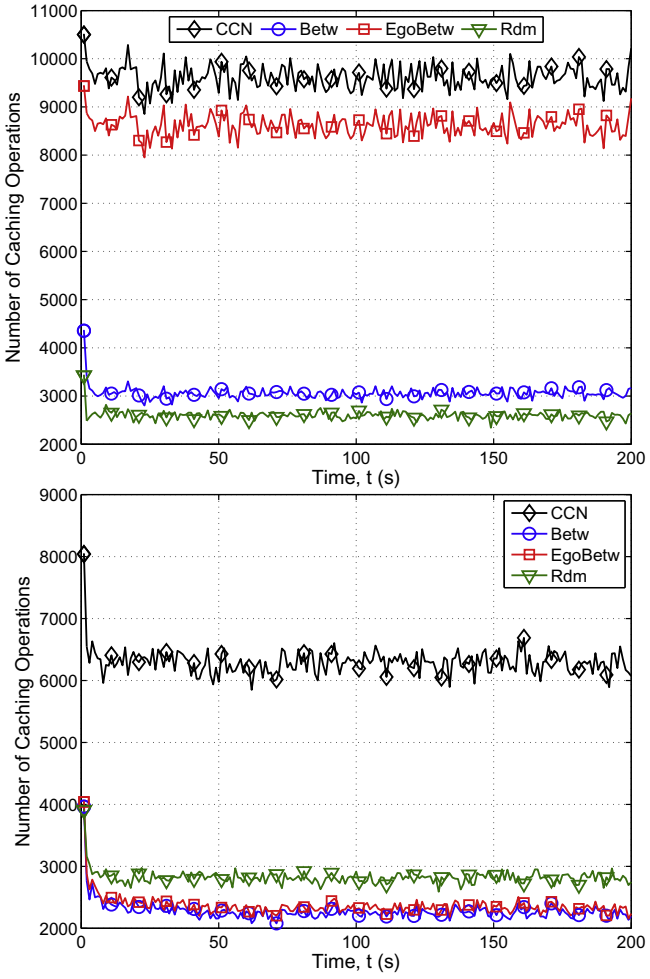

Fig. 10. Number of caching operations for the different caching schemes; (top) *k-ary* tree, (bottom) B-A graph.

the notion that B-A graphs reflect better real network topologies. These results further confirm the validity of our centrality-based caching scheme even in large-scale real network topologies. We provide in Table 3 the mean and variance of β and γ at stable operation phase (*i.e.*, after the initial transient phase).

5.5. Caching operation overhead

From the above, we have shown the consistent gain of *(Ego) Betw* in terms of cache hit and server hit. In this section, we provide an illustration on how much less caching is done to achieve this gain. We measure the number of caching operations (*i.e.*, the actual process of storing and evicting content in the nodes). Fig. 10 shows the recorded number of caching operations for the different caching schemes in a *k-ary* tree with $k = 2$ and $D = 5$ and a B-A graph with $N = 100$.

CCN always has the highest number of caching operations compared to the others since it caches non-selectively in every node along the delivery path. Also, it is apparent that the caching operation of *CCN* is highly dependent on the length of the content delivery paths. We tracked the path length of each content request between the content server and user and found that the average path lengths are 5.78 hops and 2.87 hops for the binary tree and B-A graph respectively. Note that the B-A graph has smaller average path length, albeit having a higher number of nodes. This is due to the existent of highly connected hubs (*i.e.*, small-world effect) in B-A graphs.

Betw, however, caches approximately 69% and 64% less than what *CCN* needed in a *k-ary* tree and B-A graph respectively. *EgoBetw* exhibits similar behavior a in B-A graph but not in *k-ary* tree. The reason for this is simply because all the nodes that are not root or leaf nodes have the same C_B values and thus, will all cache.

6. Characterization of *Betw*

In this section, we find the graph invariant that controls the effectiveness of our *Betw* caching scheme. We have, so far, loosely used the terms "regular" and "non-regular" for describing the topologies used in our studies. However, none of the graphs we considered is strictly regular. Thus, we require a more formal property to characterize the use of betweenness in the caching scheme. En route caching schemes are sensitive to the detail of the network structure and thus, a specific index describing the topology as a whole (*e.g.*, graph diameter, spectral radius *etc.*) cannot predict their effectiveness. Our investigation has revealed that instead of the more commonly used degree distribution (*e.g.*, used in the classification of network types [44,45]), our *Betw* caching scheme is dependent on the betweenness distribution. In the remainder of this section, we show that (1) our *Betw* scheme will better the performance of *CCN* as long as the betweenness distribution follows a power law, $P_B \sim k^{-c}$ and (2) the degree distribution of the topology does not directly affect the performance.

For the overlay network of the union of all shortest paths, [39] has analytically found that the betweenness distribution follows an inverse power law for *k-ary* trees and inverse square power law for scale-free trees. Meanwhile, [37,46] have empirically verified that the betweenness distribution for scale-free graphs and Internet AS networks follow power law. We show the pdf, cdf and rank (*i.e.*, log–log plot of sorted values) of betweenness and ego network betweenness distributions for a sample *5-ary* tree and 100-node B-A graph used in our study in Figs. 11 and 12

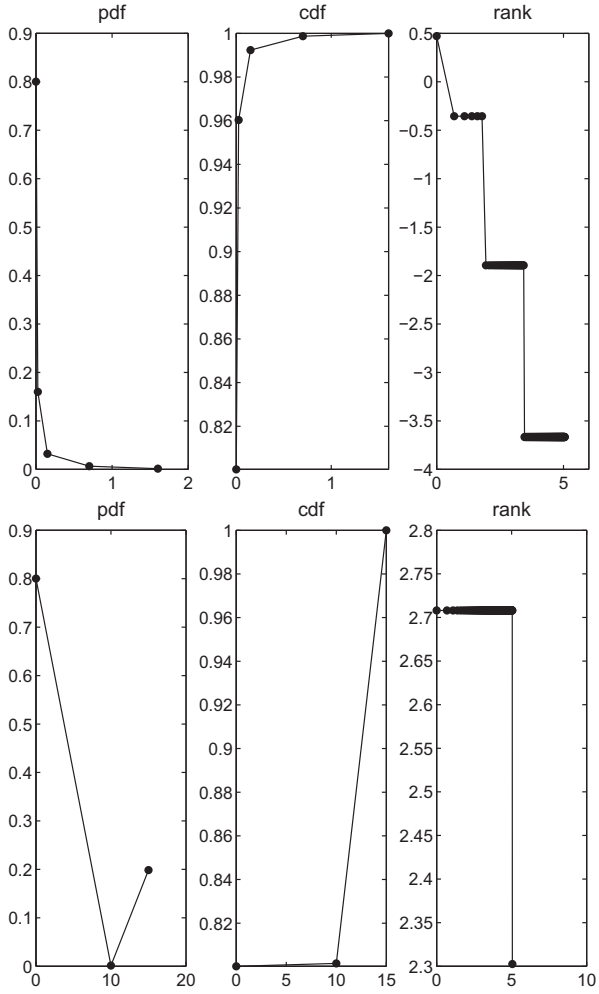

Fig. 11. 5-ary tree; (top) Betweenness, (bottom) Ego network betweenness.

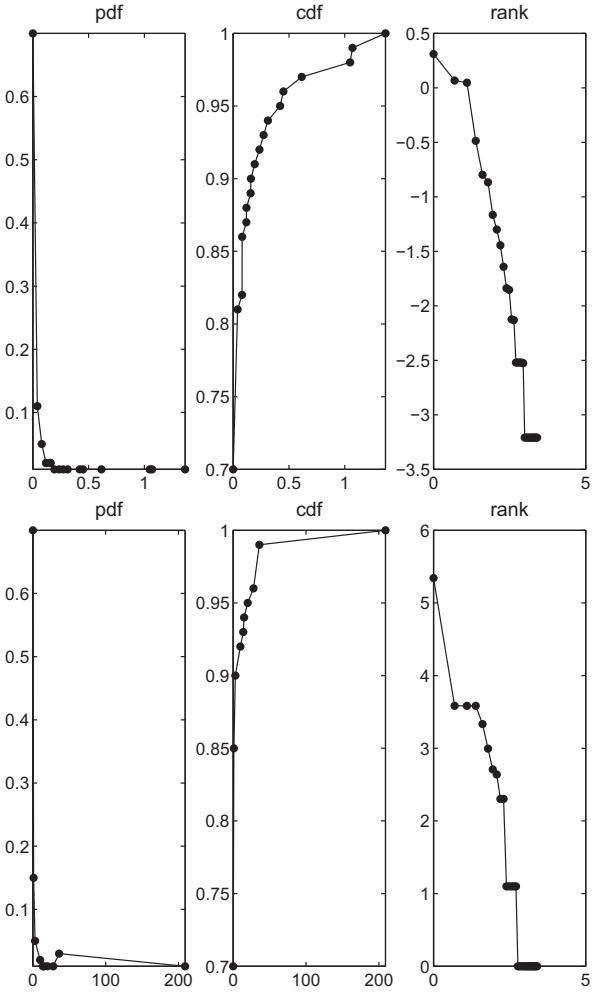

Fig. 12. B-A graph; (top) Betweenness, (bottom) Ego network betweenness.

respectively and observe agreement on the power law property of the graphs with previous findings.

In addition to that, rank plots also graphically explain the reason on the ineffectiveness of *EgoBetw* scheme in *k-ary* trees. While there are D levels of betweenness values, the ego network betweenness values are always only two (root and leaf nodes constitute one level while the rest of the nodes form the other). As such, regardless of D and k of the tree, the *EgoBetw* scheme will always approximate *CCN* in a *k-ary* topology.

We leverage the simplicity and tractability properties of the Erdős-Rényi (ER) graph model to test the effectiveness of our *Betw* caching scheme relative to the baseline *CCN* approach under different betweenness distributions. In the ER model, for a given number of nodes, a link randomly connects a pair of nodes with probability p_r independent of all other links. In our experiment, we construct a set of ER graphs with $N = 100$ nodes and $p_r = 0.05$. The value of p_r is specifically chosen to be above the sharp threshold for connectedness (*i.e.*, $p_r > p_r^c = ln(N)/N$) to ensure fully connected graphs while at the same time sufficiently small (link density = $\frac{L}{\binom{N}{2}} = p_r$ where L is the number of links) to avoid a highly meshed topology; this is because caching is of little significance in such cases because most nodes can reach each other directly without intermediate nodes. For instance, in the extreme case of a fully meshed topology, caching is redundant as it will provide no gain but instead will incur cost (*e.g.*, DRAM).

Next, we evaluate both *Betw* and *CCN* schemes for these ER graphs with edge disorder (weighted network scenario) where non-uniform link weights are applied to the links independent of the degrees of the vertices involved. By using the same set of graphs, we fix the degree distribution for each one while by varying the disorder regime (through altering the link weight distribution), we obtain different betweenness distributions for the same graph.

Specifically, we consider non-negative independent and identically distributed (i.i.d) link weights in an additive setup to create different betweenness distributions in the same graph by controlling the disorder of the graph. For this purpose, we follow [47] (Chapter 16) to use the polynomial link weight distribution

$$F_w(x) = x^{\alpha}\mathbf{1}_{x\in[0,1]} + \mathbf{1}_{x\in(1,\infty)}, \quad \alpha > 0 \tag{4}$$

where $\mathbf{1}_x$ equals one if x is true and zero otherwise. When $\alpha \to 0$, the content delivery paths are mainly determined by the highest link weight of the constituent links. This corresponds to a strong disorder limit. In this disorder regime, each path between two nodes is characterized by the maximum link weight along that path and the shortest path is simply the path with the minimal maximum link weight between the two nodes. To create a weak disorder limit, we simply revert to constant link weights (*i.e.*, non-weighted networks). In a weakly disordered system, most, if not all, of the links in a path contribute to the determination of the shortest path between two nodes.

Essentially, changing the link weight distribution results in a different set of shortest paths for the graph and thus a different

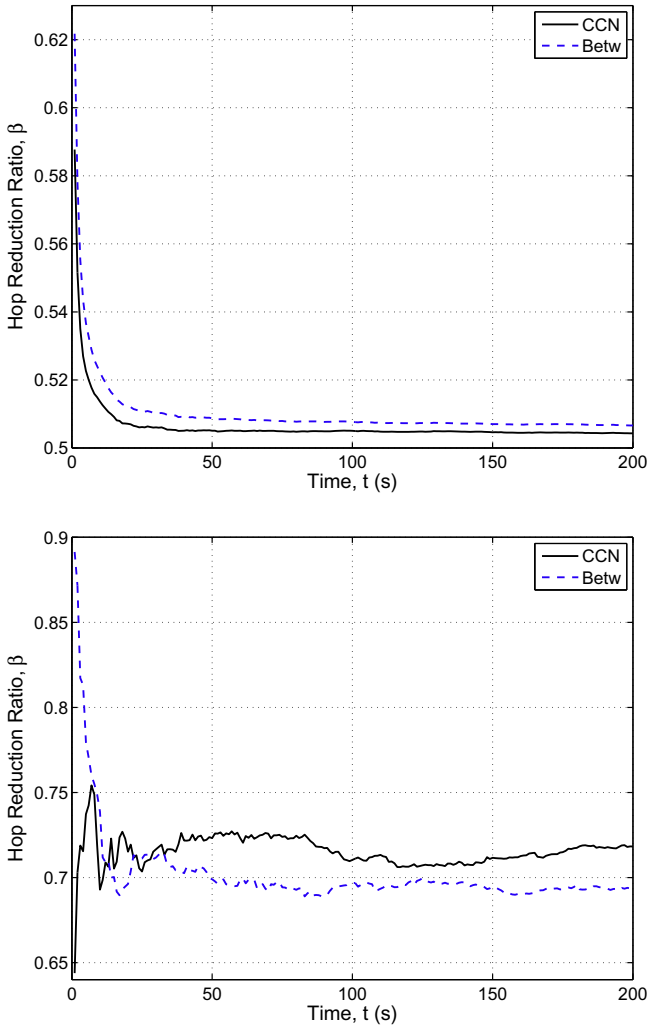

Fig. 13. β for both weak disorder (top) and strong disorder (bottom) limits.

disorder limit is achieved. The aim of creating the strong/weak disorder is to obtain different betweenness distributions for the same graph so that we can test the effectiveness of *Betw* independently of the degree distribution. Note that the betweenness of a node is directly related to the paths involving that node. The relationship between link weights and betweenness is outside the scope of this paper. Readers can find related discussions in [39,46]. The hop reduction ratio obtained for both disorder regimes are shown in Fig. 13.

It has been known that ER networks exhibit exponentially decaying betweenness distribution in the weak disorder limit and power law betweenness distribution in the strong disorder limit [37]. From the results, we found that *Betw* performs on a similar level to *CCN* when the betweenness distribution does not follow a power law (weak disorder limit) but performs better than the *CCN* when the betweenness distribution follows a power law (strong disorder limit). This result is consistent with the whole set of ER graphs we generated. Coupled with the evidence provided in Section 5, we conclude that the power law betweenness distribution ensures the effectiveness of *Betw*.

Furthermore, because of the fact that the different performances are obtained in the same ER graph, we also conclude that the degree distribution does not directly determine the effectiveness of our *Betw* scheme. This further implies that the caching performance is influenced by the delivery paths rather than the degree of the nodes involved in the delivery paths.

7. Discussions

7.1. Variants

In this section, we discuss two simple variants to our *Betw* scheme which intuitively at first glance may be able to enhance further the performance of *Betw*. However, as we have shown throughout this paper, in terms of caching in ICN, the caching gain does not increase with the number of times a content is cached along the delivery path.

First, instead of only the node with the highest centrality, we can consider to relax it so that more than one top ranked nodes cache the content (*e.g.*, the two nodes having the highest C_B values cache the content). The implication to the caching operation is that both the request and the content packets have to record and carry more C_B values and thus incur higher overhead (more control information).

A second variant to this is the use of a threshold. Any node having its C_B greater than the threshold caches the content it forwards. In this case, if a delivery path involves many highly central nodes, then the content will be cached more times. However, the tuning of the threshold is not straightforward, especially when multiple domains are involved. In fact, agreement will have to be reached between different domains.

In both variants, the resulting effect is simply an increment to the number of nodes a content is cached per request. As we have shown in our results, counter-intuitively, caching more does not necessarily translate to better gain. In fact, if we extrapolate to the extreme case where the number of times a content is cached per delivery is equal to or higher than the diameter of the topology, the performance of both variants will be simply reduced to the same performance as *CCN* which is worse than *Betw*.

7.2. Business model implications

Realising ICN and its corresponding caching paradigm in the real world will involve some fundamental changes to today's business model [48]. Currently, content storage in a domain is mainly managed by CDN companies, leaving ISPs to passively transfer bits from one point to the other. ISPs have no control or power over what is stored/cached within their own domain. In-network caching inherently changes the current state of affairs, providing ISPs with the advantage of managing what is to be cached in their own network. This might call for a change in the business relationships between content providers, CDN operators and ISPs. Although this is the case with any in-network caching approach, here we discuss associated aspects that will potentially affect business relationships among Internet players when implementing the proposed centrality-based approach to in-network caching.

The proposed centrality-based caching scheme can be implemented to indiscriminately support all content that traverse a domain, or could be implemented for selected content only. In the former case, we do not expect that business relationships between Internet players will be altered. In the latter case, decisions could be based on the ISP's own view of which content are or expected to become popular. In a similar fashion to the first case, where ISPs cache indiscriminately all content, business relationships between the ISP and respective content providers stay intact.

The rules, however, may have to change if the ISP's decisions on what to cache are based on agreements between the ISPs and content providers. That is, if content providers request (and pay for) caching of specific content within a domain, then this content would have to be explicitly tagged and given priority at the highest centrality node. In turn, this could trigger a chain of changes with respect to the contracts between the content provider and the CDN

operator. For example, to maintain its revenue, the content provider would seek to reduce the amount of money it is paying to the CDN operator. We expect that this will be the case for any proposed in-network caching scheme; we also expect that the billing schemes will be mainly based on the performance of the caching scheme in question.

However, the exact benefits for each Internet market player, as well as the detailed rules that would regulate this market are a subject of a different study. We note that in-network caching and ICN in general calls for a significant re-examination of the current Internet business models.

7.3. Misbehaving nodes

Since both the proposed centrality-based caching schemes are dependent on the C_B values, concerns may be raised regarding the consequences if these values are tinkered. First of all, we note that such a situation should not happen for a network under the administration of a single operator since all the routers are owned and controlled by the same owner. A compromised node should be easily detected. However, infrastructureless wireless networks may be more susceptible to such misbehavior.

Two misbehavior possibilities are foreseen:

- Low C_B – a node may artificially advertise a very low value of C_B (*e.g.*, 0.0) to avoid caching any content. In this case, this misbehaving node will simply be seen as a non-participating node in the network. It will enjoy the benefit of in-network caching provided by others but not contribute to the enhancement of the content delivery (parasitic behavior). However, such nodes do not affect the overall performance of the centrality-based caching schemes. In fact, we expect some non-conforming nodes to exist in the network (especially during the early deployment phase of ICN).
- High C_B – a node may artificially advertise a very high value of C_B. In this case, this misbehaving node will be selected to cache all the content that traverses it. If this misbehaving node does not cache the content, then it nullifies the caching gain of all the content delivery paths that involve the said node. Paths that do not include this node will not be affected. Also, the actual content delivery will still operate as normal.

For *Betw*, a misbehaving node can easily be detected since the C_B values are centrally computed. For *EgoBetw*, a possible way to detect such a node is to have the neighboring node(s) cross-check with the target's node's neighbor(s) since each node has to broadcast its list of immediate neighbors during the C_B computation phase. If no response / acknowledgment is received from the neighbors, then it is clear that the node is attempting to inflate its C_B by advertising non-existent neighbors. The detailed security mechanism and its implications (*e.g.*, signalling overhead for detecting the malicious nodes) is outside the scope of this paper and will be explored in our future work.

8. Summary and conclusions

We argue against the necessity of an indiscriminate in-path caching strategy in ICN and investigate the possibility of caching less in order to achieve higher performance gain. We first demonstrated that a simple random caching strategy (*Rdm*) can outperform (though inconsistently) the current pervasive caching paradigm under the conditions that the network topology has low number of distinct content delivery paths and high average delivery path length. We, then, proposed a caching strategy based on the concept of betweenness centrality (*Betw*) such that content is only cached at the nodes having the highest probability of getting a cache hit along the content delivery path. Our design ensures the content always spreads towards content users and thus reduces the content access latency. We also proposed an approximation of it (*EgoBetw*) for scalable and distributed realization in dynamic network environments where the full topology cannot be known *a priori*.

We compared the performance of our proposals against the ubiquitous caching of the CCN proposal [2] (*CCN*). Based on our extensive simulations, we observed that *Betw* consistently achieves the best hop and server reduction ratios across topologies having different structural properties without being restricted by the operating conditions required by *Rdm*. Our results further suggest that *EgoBetw* approximates closely *Betw* in non-regular topologies (*e.g.*, B-A graphs) and thus presents itself as a practical candidate for the deployment of this approach. Besides synthetic topologies (*i.e.*, *k-ary* trees and B-A graphs), the observations were further verified through a large-scale real Internet topology. We also showed that the caching overhead of *CCN* is more than 60% higher than our *Betw*. Thus, we conclude that indeed caching less can actually achieve more and that our proposed *(Ego)Betw* approach is a potential candidate for realizing this promise. In our investigation we also found that the caching schemes tend to be sensitive to the detail of network structure. We identified that the effectiveness of *Betw* is dependent on the betweenness distribution and found that topologies that exhibit power law distribution (*e.g.*, Internet AS and WWW networks) ensure its effectiveness.

References

[1] T. Koponen et al., A data-oriented (and beyond) network architecture, in: Proc. ACM SIGCOMM, Kyoto, Japan, Aug. 2007.
[2] V. Jacobson et al., Networking named content, in: Proc. ACM CoNEXT, 2009, pp. 1–12.
[3] D. Trossen et al., Conceptual Architecture: Principles, Patterns and Subcomponents Descriptions, May 2011. <http://www.fp7-pursuit.eu/PursuitWeb/>.
[4] P. Jokela et al., LIPSIN: line speed publish/subscribe inter-networking, in: Proc. ACM SIGCOMM, Barcelona, Spain, 2009.
[5] G. Garcia et al., COMET: Content mediator architecture for content-aware networks, in: Proc. of the Future Network and Mobile Summit 2011, Warsaw, Poland, IEEE, June 2011.
[6] G. Pavlou, N. Wang, W.K. Chai, I. Psaras, Internet-scale Content Mediation in Information-centric Networks, Ann. Telecommun., Special Issue on Networked Digital Media, Springer, in press. http://dx.doi.org/10.1007/s12243-012-0333-8.
[7] W.K. Chai et al., CURLING: content-ubiquitous resolution and delivery infrastructure for next-generation services, IEEE Commun. Mag. 49 (3) (2011) 112–120.
[8] L. Rizzo, L. Vicisano, Replacement policies for a proxy cache, IEEE/ACM Trans. Netw. 8 (2) (2000) 158–170.
[9] P.B. Danzig, R.S. Hall, M.F. Schwartz, A case for caching file objects inside internetworks, in: ACM SIGCOMM, Sept. 1993, pp. 239–243
[10] P. Krishnan, D. Raz, Y. Shavitt, The cache location problem, IEEE/ACM Trans. Netw. 8 (5) (2000) 568–582.
[11] I. Psaras, R. Clegg, R. Landa, W.K. Chai, G. Pavlou, Modelling and evaluation of CCN-caching trees, in: Proc. of IFIP Networking, Valencia, Spain, May 2011.
[12] W.K. Chai, D. He, I. Psaras, G. Pavlou, Cache less for more in information-centric networks, in: Proc. of IFIP Networking Prague, Czech Republic, May 2012.
[13] G. Carofiglio, M. Gallo, L. Muscariello, D. Perrino, Modelling data transfer in content centric networking, in: Proc. International Teletraffic Congress (ITC), 2011.
[14] S. Arianfar, P. Nikander, J. Ott, Packet-level caching for information-centric networking, Finnish ICT-SHOK Future Internet Project, Tech. Rep., 2010.
[15] A. Dan, D. Towsley, An approximate analysis of the lru and fifo buffer replacement schemes, in: ACM SIGMETRICS, 1990.
[16] P. Jelenkovic, A. Radovanovic, M.S. Squillante, Critical sizing of lru caches with dependent requests, J. Appl. Probab. 43 (4) (2006) 1013–1027.
[17] N. Laoutaris et al., Distributed selfish caching, IEEE Trans. Parallel Distrib. Syst. 18 (10) (2007).
[18] G. Dán, Cache-to-cache: could ISPs cooperate to decrease peer-to-peer content distribution costs? IEEE Trans. Parallel Distrib. Syst. 22 (9) (2011).
[19] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: modelling, design and experimental results, IEEE J. Sel. Areas Commun. 20 (7) (2002).
[20] N. Laoutaris, H. Che, I. Stavrakakis, The LCD interconnection of LRU caches and its analysis, Perform. Eval. 63 (7) (2006) 609–634.

[21] A. Heddaya, S. Mirdad, Webwave: globally balanced fully distributed caching of hot published documents, in: Proc. IEEE Int'l. Conf. Distributed Computing Systems, Baltimore, MD, May 1997, pp. 160–168.
[22] L. Zhang, S. Floyd, V. Jacobson, Adaptive web caching, in: NLANR Web Cache Workshop, June 1997.
[23] L. Zhang, Adaptive web caching: toward a new global caching architecture, in: 3rd Int'l World Wide Web Caching Workshop, Manchester, UK, June 1998.
[24] E.J. Rosensweig, J. Kurose, D. Towsley, Approximate models for general cache networks, in: IEEE INFOCOM, 2010.
[25] D. Rossi, G. Rossini, Caching performance of content centric networks under multi-path routing, Technical Report, 2011.
[26] C. Fricker, P. Robert, J. Roberts, N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, in: IEEE INFOCOM NOMEN Workshop 2012.
[27] D. Rossi, G. Rossini, On sizing CCN content stores by exploiting topological information, in: IEEE INFOCOM NOMEN Workshop 2012.
[28] G. Tyson et al., A trace-driven analysis of caching in content-centric networks, in: Proc. 21st Int'l. Conf. on Comp. Commun. Networks (ICCCN), Germany, 2012.
[29] K. Cho et al., WAVE: popularity-based and collaborative in-network caching for content-oriented networks, in: IEEE INFOCOM NOMEN Workshop 2012.
[30] I. Psaras, W.K. Chai, G. Pavlou, Probabilistic in-network caching for information-centric networks, in: ACM SIGCOMM ICN Workshop 2012.
[31] Wang et al., Advertising cached contents in the control plane: necessity and feasibility, in: IEEE INFOCOM NOMEN Workshop 2012.
[32] A. Ghodsi, Information-centric networking: seeing the forest for the trees, in: ACM Workshop on Hot Topics in Networks (HotNets-X), Cambridge, MA, Nov. 2011.
[33] T.M. Wong, J. Wilkes, My cache or yours? Making storage more exclusive, in: Proc. USENIX Annual Technical Conference, Monterey, CA, 2002, pp. 161–175.
[34] M. Zink, K. Suh, Y. Gu, J. Kurose, Watch global, cache local: YouTube network traces at a campus network – measurements and implications, in: Proc. IEEE Multimedia Computing and Networking, 2008.
[35] S. Wassermann, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, Cambridge, 1994.
[36] K.-I. Goh, B. Kahng, D. Kim, Universal behavior of load distribution in scale-free networks, Phys. Rev. Lett. 87 (27) (2001) 31.
[37] K.-I. Goh et al., Load distribution in weighted complex networks, Phys. Rev. E 72 (2005) 017102.
[38] A. Beben, J. Mongay Batalla, W.K. Chai, J. Śliwiński, Multi-criteria decision algorithms for efficient content delivery in content networks, Annals of Telecommunications, Special Issue on Networked Digital Media, Springer, in press. http://dx.doi.org/10.1007/s12243-012-0321-z.
[39] H. Wang, J.M. Hernandez, P. Van Mieghem, Betweenness centrality in a weighted network, Phys. Rev. E 77 (2008) 046105.
[40] M. Everett, S. Borgatti, Ego network betweenness, Social Networks 27 (2005) 31–38.
[41] P. Pantazopoulos, M. Karaliopoulos, I. Stavrakakis, Centrality-driven scalable service migration, in: Proc International Teletraffic Congress (ITC), 2011.
[42] A.L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[43] CAIDA dataset. <http://www.caida.org/research/topology/Datasets>.
[44] L.A. Amaral, A. Scala, M. Barthélémy, H.E. Stanley, Classes of small-world networks, Proc. Nat. Acad. Sci. U.S.A. 97 (2000) 11149.
[45] K.-I. Goh et al., Classification of scale-free networks, Proc. Nat. Acad. Sci. U.S.A. 99 (2002) 12583–12588.
[46] R. Pastor-Satorras, A. Vespignani, Evolution and Structure of the Internet: A Statistical Physics Approach, Cambridge University Press, Cambridge, 2004.
[47] P. Van Mieghem, Performance Analysis of Communications Systems and Networks, Cambridge University Press, Cambridge, 2006.
[48] W.K. Chai, M. Georgiades, S. Spirou, Towards information-centric networking: research, standardization, business and migration challenges, in: H. Moustafa, S. Zeadally (Eds.), Media Networks: Architectures, Applications, and Standards, 2012, CRC Press, Taylor & Francis Group.