Cache "Less for More" in Information-Centric Networks

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

Cache "Less for More" in Information-centric Networks

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

Department of Electronic and Electrical Engineering,
University College London,
WC1E 6BT, Gower Street, London, UK
{w.chai, diliang.he.10, i.psaras, g.pavlou}@ee.ucl.ac.uk

Abstract. Ubiquitous in-network caching is one of the key aspects of information-centric networking (ICN) which has recently received widespread research interest. In one of the key relevant proposals known as Networking Named Content (NNC), the premise is that leveraging in-network caching to store content in every node it traverses along the delivery path can enhance content delivery. We question such indiscriminate universal caching strategy and investigate whether caching *less* can actually achieve *more*. Specifically, we investigate if caching only in a subset of node(s) along the content delivery path can achieve better performance in terms of cache and server hit rates. In this paper, we first study the behavior of NNC's ubiquitous caching and observe that even naïve random caching at one intermediate node within the delivery path can achieve similar and, under certain conditions, even better caching gain. 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.

Keywords: Information-centric networking, caching, betweenness centrality.

1 Introduction

Information-centric networking (ICN) has recently attracted significant attention, with various research initiatives (e.g., DONA [1], NNC [2], PSIRP/PURSUIT [3][4] and COMET [5]) 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 used for content access and delivery, with a high volume of digital content (e.g., 3D/HD movies, photos etc.) delivered to users who are only interested in the actual content rather than the source location. As such, we no longer need a natively supported content distribution framework. 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 element, avoiding going all the way to the hosting server. Out of the current ICN approaches, NNC [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. 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 to cache and how can we choose them?

Our contribution in this study is three-fold. 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.

In the next section, we define the system of interest and layout our arguments and rationale with a motivating example illustrating that caching *less* can be *more*. We then describe 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 [6][7][8]), with the content source(s) 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 scale-free topologies whose properties imitate closely the real Internet topology.

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

2 Caching in ICN

2.1 Model Description and Problem Statement

As a foundation, we first assume that the network has an ICN publish/subscribe framework in place (e.g., [1][2][3][4][5]). Specifically, we assume that a content request and resolution mechanism is already in place. As pointed out in [9], all the different ICN proposals in the literature invariably have such common functions (although with different primitives). Let $G = (V, E)$ be an undirected network with $V = \{v_1, \dots, v_N\}$ nodes and $E = \{e_1, \dots, e_M\}$ links. We denote $F = \{f_1, \dots, f_R\}$ the content population in the system and $S = \{s_1, \dots, s_P\}$ 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, $1 \leq r \leq R$, 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 more sophisticated caching algorithms for achieving better gain.

2.2 Related Work and Motivation

In the networking area, caching has been studied in standalone caches [10][11] focusing on the performance of different cache replacement policies. 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 [12][13]) or structured (e.g., hierarchical caching [14][15]) caching approaches being considered. In ICN, caching takes place within the network, requiring line-speed operation; in this context, complex algorithms executed by multiple collaborating entities that require information exchanges are simply not feasible. One of the key ICN proposals, networking named content (NNC) [2], defines its ubiquitous caching 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 *NNC+LRU* and treat it as the benchmark for performance comparison.

Such a ubiquitous caching scheme has already raised doubts (e.g., [9]). In the general cache-related literature, some authors have already questioned this aggressive "cache-everything-everywhere" strategy [14][15][16]. 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+LRU*, 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 $s_1, P = 1$, is located at v_1 (root) while content requests originate exogenously from other nodes. 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 (*NNC+LRU*).

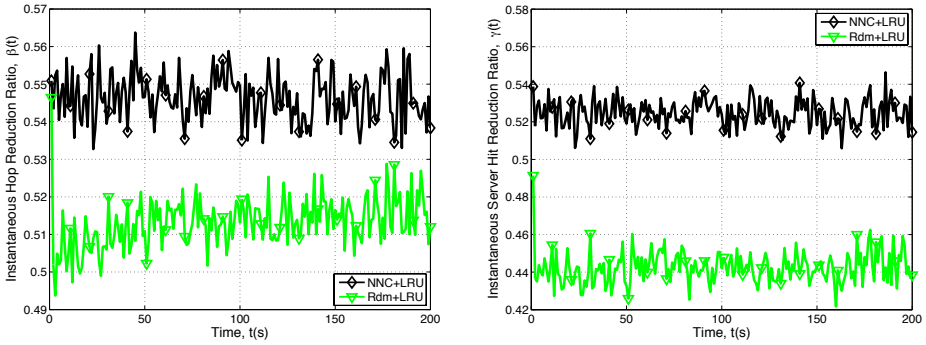

Fig. 1. Simple random caching outperforming ubiquitous caching in the number of hops to hit the content (left) and reduced server hits (right).

3 A Centrality-based Caching Scheme

3.1 Basic Algorithm

Based on the above observations, we realize that caching indiscriminately does not necessarily guarantee the highest cache hit rate. On the other hand, this result cannot be used as conclusive evidence that caching less is better since the string topology constrains to 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 hit. Following this argument, we propose a novel caching scheme based on the concept of betweenness centrality [17] which measures the number of times a specific node lies on the content delivery path between all pairs of nodes in a network topology. The basic idea is that if a node lies along a high number of content delivery paths, then it is more likely to get a cache hit. By caching only at those more "*important*" nodes, we reduce the cache replacement rate while still caching content where a cache hit is most probable 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 *NNC+LRU*, all four nodes will retain a copy of the content while under *Rdm+LRU*, only one of them will cache the content. Let's assume now that client B requests the same content. For *NNC+LRU*, 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+LRU*, there is ¼ chance to get a cache miss (i.e., content cached at v_4) and ½ chance that the hop count reduction is worse than *NNC+LRU* (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).

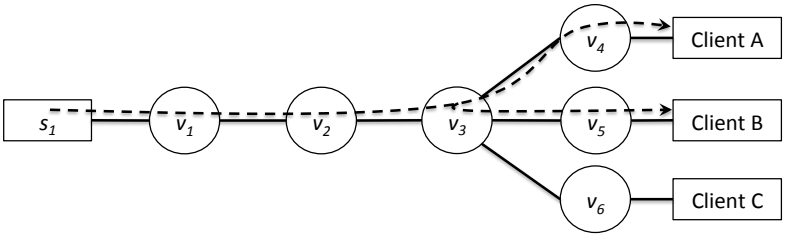

Fig. 2. An example topology with optimal caching location at v_3.

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

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

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. Computation of the delivery paths can be done online or dynamically, as our scheme does not require *a priori* path knowledge. Without loss of generality, we use the shortest path as the content delivery path in this paper.

Betw+LRU 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 [5]) records the highest centrality value among all the intermediate nodes. It may be inserted into the request packet header. This value is copied onto the content messages during the data transmission at the server. 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 independently makes its caching decision 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 given below:

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
```

3.2 Approximation via Distributed Computation

We now sketch a distributed implementation for *Betw+LRU* where the full network topology may not be readily available because of an infrastructure-less network with relatively dynamic topology (e.g., for 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 *approximation* of their C_B. This approximation is based on the ego network betweenness concept [18]. The ego network consists of a node together with all of its immediate neighbours 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 [18], 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 the ego network betweenness is $A^2[\mathbf{1} - A]_{i,j}$ where $\mathbf{1}$ is a matrix of 1's.

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 neighbours with message Time-To-Live=1 when it first joins the network and whenever there are changes to its one-hop neighbour set. The overhead is thus limited as the message propagation is limited to one hop only. 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 $\sigma_{i,j}(v)$ of v's ego network. The rest of the caching operations remain unchanged (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 [19]. Coupled with its low computation complexity (reduced from $\mathcal{O}(NM)$[2] to $\mathcal{O}(d_{max}^2)$ 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+LRU* hereafter. Referring back to Fig. 2, the outcome of *Betw+LRU* and *EgoBetw+LRU* is the same since v_3 remains the node having the highest centrality value.

4 Performance Evaluation

4.1 Performance Metrics and Simulation Scenarios

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).

$$Hop\ reduction\ ratio, \beta(t) = \frac{\sum_{r=1}^{R} h_r(t)}{\sum_{r=1}^{R} H_r(t)} \quad (2)$$

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, $\beta = 1.0$.

[2] 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.

$$Server\ hit\ reduction\ ratio, \gamma(t) = \frac{\sum_{r=1}^{R} w_r(t)}{\sum_{r=1}^{R} W_r(t)} \quad (3)$$

where $W_r(t)$ is the number of request for f_r from t-1 to t and $w_r(t)$ is the number of server hits for f_r from 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 $k + 1$ valence) and (2) scale-free topologies following the Barabasi-Albert (B-A) power law model [20] 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 [21].

Content requests for different content are generated based on Zipf-distribution with $\sum_{r=1}^{R}(C/r^{\alpha}) = 1$ where the probability for a request for the r^{th} popular content is C/r^{α} with α being the popularity factor. We use $\alpha = 1.0$[3] and requests originate randomly from all nodes. Each simulation run begins with all cache stores being empty (i.e., cold start). Unless otherwise specified, the simulations are run with the following parameters: total simulation time = 200 s, $\lambda = 5{,}000$ request/s, content population = 1,000 and uniform cache store size = 10% of total content population.

4.2 Experiments with *k*-ary Trees

(a) 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 depth of the tree from root. We first 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, *NNC+LRU* always reaches the stable performance level first. This is due to its *always cache* policy.

We observe that both *Betw+LRU* and *Rdm+LRU* perform better than *NNC+LRU* for both metrics. Tracking the evolution of the cache stores over time revealed that this is due to the high cache replacement rate in *NNC+LRU*. Replacing cached content rapidly causes content often being evicted before the next matching request is received. We have shown this in [6]. 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 caching "*less for more"* 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

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

selected rather than caching at nodes with high betweenness. Finally, an interesting observation is that instead of approximating the performance of the *Betw+LRU* scheme as it was meant to be, *EgoBetw+LRU* actually performs at the same level as *NNC+LRU*. This is the 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+LRU* is simply reduced to a similar behavior with *NNC+LRU*.

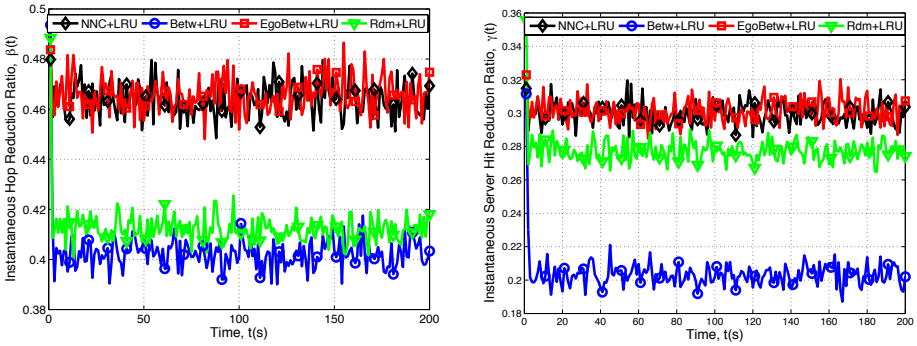

Fig. 3. Instantaneous behavior of the caching schemes for a binary tree; (left) β, (right) γ.

(b) 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.

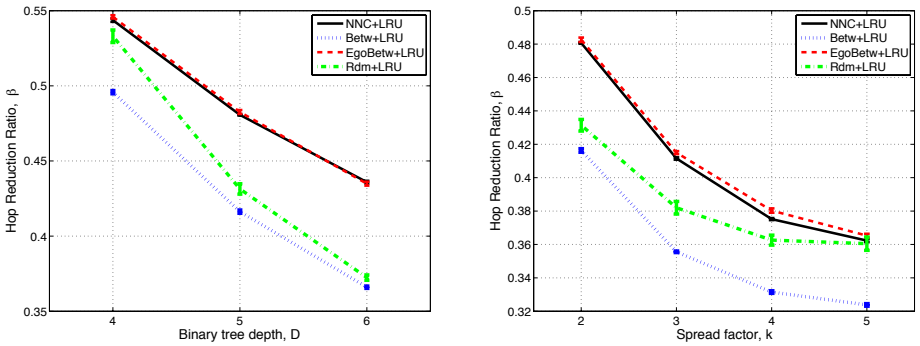

Fig. 4. *Betw+LRU* consistently outperforms the rest over different D (left) and k (right).

We find that while the performance distance between *NNC+LRU* and *Betw+LRU* remains approximately constant, *Rdm+LRU* does not exhibit such consistency.

Rdm+LRU 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 *NNC+LRU*. 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+LRU* 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 gain (reducing the content delivery hop count).

4.3 Experiments with Scale-free Topologies

(a) 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 [20]. As such, we consider scale-free topologies following the construction method described in [20] (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+LRU* and *EgoBetw+LRU*) perform better than *NNC+LRU* for both metrics and *EgoBetw+LRU* now approximates closely *Betw+LRU*. 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+LRU* algorithm can be used for irregular graphs.

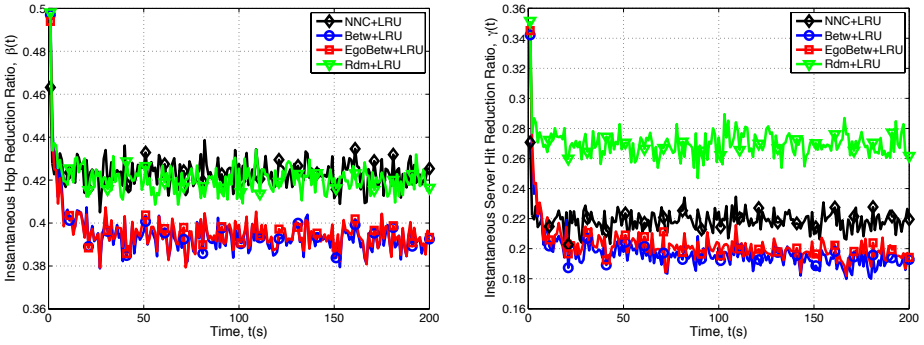

Fig. 5. Instantaneous behavior of the caching schemes in a B-A graph; (left) β, (right) γ.

Secondly, we observe that *Rdm+LRU* no longer outperforms *NNC+LRU*. In fact, it performs at the same level as *NNC+LRU* 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).

(b) 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 the probability proportional to the attractiveness of existing nodes (i.e., preferential attachment). We evaluate the caching schemes over ten B-A graphs with $N = 100$ and mean valence = 2. From Fig. 6 (left), both centrality-based caching schemes perform better than the rest. However, *Rdm+LRU* is worse than *NNC+LRU* in most 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. 6 (right) shows how ego network betweenness approximates betweenness in a B-A graph.

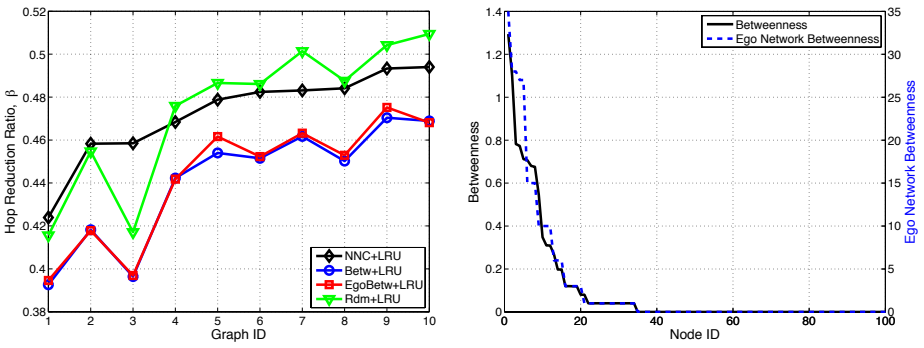

Fig. 6. Caching performance with different 100-node B-A graphs (left) and a sample ego network betweenness and betweenness values of the nodes in a B-A graph (right).

From Fig. 7 (left), we observe again that centrality-based caching schemes provide the best hop reduction ratio while *Rdm+LRU* exhibits inconsistent gain across B-A graphs with different sizes. We observe that as the size of the topology increases, *Rdm+LRU* gradually performs worse than *NNC+LRU*. 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+LRU* does not differentiate the centrality of the nodes, there is higher probability of *Rdm+LRU* 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. 7 (right), 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., Fig. 1, 3 and 5).

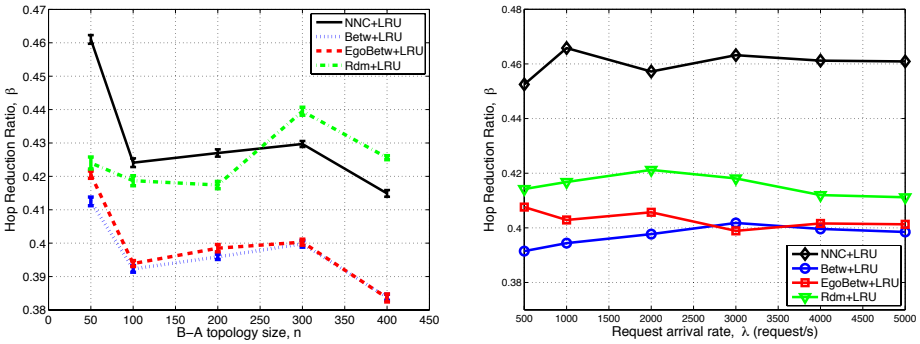

Fig. 7. Hop reduction ratio for different B-A graph sizes (left) and request rates, λ (right)

In Table 1, 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+LRU* reliably achieves better gains (both in terms of hop and server hit reduction) in comparison to *NNC+LRU*. For instance, it reduces server hits over 30% and hop count over 17% in comparison to *NNC+LRU* in the string topology.

Table 1. 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)	
	$\sum h$	$\sum w$	$\sum h$	$\sum w$	$\sum h$	$\sum w$
NNC+LRU	2,6839,45	498,603	2,684,325	299,657	2,137,015	211,852
Betw+LRU	2,211,248	337,362	2,331,061	203,673	2,045,852	204,479
EgoBetw+LRU	2,680,614	497,146	2,698,153	301,797	2,074,089	207,628
Rdm+LRU	2,206,002	377,289	2,386,569	277,575	2,195,303	291,560

4.4 Experiments with the Real Internet 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 [22]. The topology is rooted at a tier-1 ISP (AS7018) and contains 6804 domains and 10205 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 8 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 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 real network topologies.

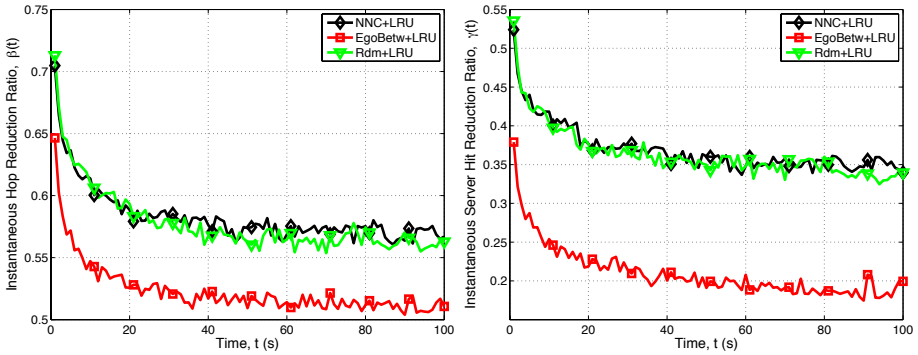

Fig. 8. Instantaneous behavior of the caching schemes in a large-scale real Internet topology; (left) β, (right) γ.

5 Summary and Conclusions

We argue against the necessity of a ubiquitous 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+LRU*) 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+LRU*) such that content is only cached at the nodes having the highest probability of getting a cache hit along the content delivery path. We also proposed an approximation of it (*EgoBetw+LRU*) 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 NNC proposal [2] (*NNC+LRU*). Based on our extensive simulations, we observed that *Betw+LRU* 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+LRU*. Our results further suggest that *EgoBetw+LRU* approximates closely *Betw+LRU* 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 are further verified with a large-scale real Internet topology. Thus, we conclude that indeed caching *less* can achieve *more* and our proposed *(Ego)Betw+LRU* is a candidate for realizing this promise.

Acknowledgements. This work was undertaken under the Information Society Technologies (IST) COMET project, which is partially funded by the Commission of the European Union. We would also like to thank our project partners who have implicitly contributed to the ideas presented here.

References

1. T. Koponen, et. al., "A Data-oriented (and Beyond) Network Architecture," in *Proc. ACM SIGCOMM '07*, Kyoto, Japan, Aug. 2007.
2. V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, R. L. Braynard, "Networking Named Content," *Proc. ACM CoNEXT*, 2009, pp.1-12.
3. D. Trossen et al., "Conceptual Architecture: Principles, Patterns and Sub-components Descriptions", May 2011. http://www.fp7-pursuit.eu/PursuitWeb/
4. P. Jokela, A. Zahemszky, C. Rothenberg, S. Arianfar and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-networking", *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009.
5. W. K. Chai, et. al., "CURLING: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 112-120, 2011
6. I. Psaras, R. Clegg, R. Landa, W. K. Chai and G. Pavlou, "Modelling and evaluation of CCN-caching trees," *Proc. of IFIP NETWORKING*, Valencia, Spain, May 2011.
7. G. Carofiglio, M. Gallo, L. Muscariello and D. Perrino, "Modelling data transfer in content centric networking," *Proc. International Teletraffic Congress (ITC)*, 2011.
8. S. Arianfar, P. Nikander, and J. Ott, "Packet-level caching for information-centric networking," Finnish ICT-SHOK Future Internet Project, Tech. Rep., 2010.
9. A. Ghodsi, et. al., "Information-centric Networking: Seeing the forest for the trees", *ACM Workshop on Hot Topics in Networks (HotNets-X)*, Cambridge, MA, Nov. 2011.
10. A. Dan and D. Towsley, "An approximate analysis of the lru and fifo buffer replacement schemes," *ACM SIGMETRICS*, 1990, pp. 143-152.
11. P. Jelenkovic, A. Radovanovic and M. S. Squillante, "Critical sizing of lru caches with dependent requests," *Journal of Applied Probability*, vol. 43, no. 4, pp. 1013-1027, 2006.
12. N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, "Distributed selfish caching," *IEEE Trans. on Parallel and Distributed Systems*, vol. 18, no. 10, 2007.
13. G. Dán, "Cache-to-Cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?," *IEEE Trans. on Parallel and Distributed Systems*, vol. 22, no. 9, 2011.
14. H. Che, Y. Tung, Z. Wang, "Hierarchical web caching systems: modelling, design and experimental results," *IEEE Journ. on Selected Areas of Communications*, 20(7), 2002.
15. 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, July 2006.
16. T. M. Wong, J. Wilkes, "My cache or yours? Making storage more exclusive," *Proc. USENIX Annual Technical Conference*, Monterey, CA, 2002, pp. 161-175.
17. L. R. Izquierdo, R. A. Hanneman, "Introduction to the Formal Analysis of Social Networks Using Mathematica", University of California, Riverside.
18. M. Everett, S. Borgatti, "Ego network betweenness", Social Networks, 27(2005) pp. 31-38.
19. P. Pantazopoulos, M. Karaliopoulos, I. Stavrakakis, "Centrality-driven scalable service migration," *Proc. International Teletraffic Congress (ITC)*, 2011.
20. A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509-512, Oct. 1999.
21. H. Wang, J. M. Hernandez and P. Van Mieghem, "Betweenness centrality in a weighted network," *Physical Review* E 77, 046105, 2008.
22. CAIDA dataset; http://www.caida.org/research/topology/#Datasets