On-Path Caching: The Benefits of Popularity

Andriana Ioannou & Stefan Weber
School of Computer Science and Statistics
Trinity College
Dublin 2, Ireland
{ioannoa,sweber}@scss.tcd.ie

ABSTRACT

Information-Centric Networking (ICN), an alternative to the current Internet architecture, focuses on the distribution and retrieval of content instead of the transfer of information between specific endpoints. Approaches to ICN employ caches in the network to eliminate the transfer of information over lengthy communication paths from a source to consumers.

The contribution of this paper lies in the placement of copies in on-path in-network caching. Our goal is to investigate the suitability of a probabilistic algorithm, *Prob-PD*, based on two variables, the content's popularity rates P and the distance ratio of each node from the source D, with regard to the cache hit rates, cache replacement rates, content delivery times and hop count rates. Towards this goal, we present a detailed comparison of simulation results of the proposed caching mechanism and published alternatives based on the YouTube traffic. Our results suggest that the performance of the algorithms is affected by the catalog size |O|. In particular, our approach may provide significant gains if certain conditions are met, such as $|O| \leq 10.000$.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design, Internetworking; H.3.2 [**Information Storage and Retrieval**]: Information Storage

General Terms

Design,Performance

Keywords

Distributed networks; Future Internet Architectures; Information-Centric Networks; Caching technologies; Content replication; On-path caching

1. INTRODUCTION

Information-Centric Networking (ICN) is an alternative to the end-to-end communication paradigm based on the publish-subscribe model [10]. In ICN, content sources make their content available by publishing it to a content notification service, i.e. a name resolution service or a name-based routing service, while clients request content from a content notification service by subscribing to it. In contrast to existing content distribution technologies, Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) networks, ICN networks are free of the application restrictions and the commercial boundaries [31] that the previous ones apply.

ICN architectures identify content resources, such as web pages, files or parts of a content resource, chunks or packets, using a content identifier; object, chunk or packet naming granularity. Ideally, content identifiers should involve no information that would bind the content to a specific location [3, 32]. If this constraint is met, content can be freely replicated, therefore, provided by more than one source. Approaches to caching can be categorized into *off-path caching* and *on-path caching* with regard to the location of caches [1, 33].

Off-path caching, also referred as content replication or content storing, aims to replicate content within a network, regardless of the forwarding path. Off-path caching is usually centralized and involves a great amount of information collected as well as advertised into a content notification service. The ICN off-path caching problem is equivalent to the CDN content replication and the web cache placement problems [28, 33]. As such, existing algorithms can be used [14, 16].

On-path caching on the other hand, is integrated to the architecture itself, i.e. the caching decision is limited to the content propagated along the delivery path(s) and to the nodes lying on the delivery path(s), caching is accomplished at the network layer, on a chunk level or a packet level, thus, being independent of the application used, caching mechanisms are bounded by the on-line speed requirements of the delivery process where the overhead of monitoring, collection of statistical information or advertisement of the cached content into a content notification service may not be acceptable or feasible. Finally, on-path caching does not follow any structural model; the topology is considered arbitrary.

In this paper, we focus on the efficiency of caching mechanisms for the on-path caching problem identified in the area of ICN. Towards this goal we propose a prob-

abilistic algorithm, *Prob-PD*, based on two variables, the content's popularity ratio P, and the distance ratio of each node from the source D, which we compare against the alternatives via simulations.

The remainder of this paper is structured as follows. In section 2, we describe the related work in the area of on-path caching and conclude to the most efficient algorithms. In Section 3, we identify the gaps of the literature review and describe the *Prob-PD* algorithm. In Section 4, we provide the details of the simulation system model. In section 5, we present the evaluation results of the algorithm against the most efficient alternatives concluded in Section 2, as well as indications for future work. We close the paper with Section 6, dedicated to the conclusions.

2. RELATED WORK

A variety of probabilistic on-path caching algorithms have been proposed, according to which a node n on the delivery path, decides to cache a content based on a probability p, [2, 17]. Probabilistic algorithms can be categorized depending on the way that the caching probability is calculated; based on a fixed value [2, 9], *FIX(p)* or based on a mathematical formula [23]. In a *FIX(p)* approach, p may be decided arbitrary or determined based on the number of nodes of the delivery path [7, 9]. The last category of algorithms is called *UniCache*. The CE^2 algorithm proposed by Jacobson et al. [13], is a form of *FIX(p)* algorithm with *p=1*. *FIX(p)* algorithms introduce no overhead between the nodes, but they are unable to exploit any knowledge regarding the content or the network topology, resulting into low performance gains and high redundancy rates.

FIX(p) algorithms have been also tested against the *LCD* algorithm [25]. The *LCD* algorithm, caches a copy of the requested content one hop closer to the client each time a content request arrives. According to the results provided by Rossi et al., the *LCD* algorithm results into lower gains compared to the *FIX(p)* and CE^2 approaches with regard to the cache hit rates. Recent studies have used the *LCD* algorithm in combination with an exponentially inceasing content caching approach [9]. The examination, though, of such an algorithm is out of the scope of this paper as every on-path caching algorithm is able to be combined with it.

Psarras et al. [23] have suggested a probabilistic caching algorithm called *ProbCache*, composed by two factors, the *TimesIn* factor and the *CacheWeight* factor. The goal of the algorithm is to provide fairness regarding the capacity of the delivery path. In order to achieve that, *TimesIn* factor indicates the number of replicas expected to be cached along the delivery path, where values x and y correspond to the number of hops travelled from the requestor to the content source and from the source to the requestor, respectively. The x value remains stable during the delivery of the content. Therefore, ProbCache is calculated based on the capacity of the remaining nodes at the delivery path. According to the authors, *TimesIn* factor favors contents that travel from further away while *CacheWeight* factor acts as a counter-balance. Psarras et al. have evaluated their algorithm against the *FIX(p)*, CE^2 and *LCD* algorithms, indicating significant gains in terms of server hit rates, hop count rates and eviction rates.

Sourlas et al. [29] have proposed an on-path caching algorithm called *LeafNode*. According to this approach, content is cached at the last node of the delivery path. Sourlas et al. have evaluate the *LeafNode* algorithm against the CE^2, resulting into higher hop count rates and lower absorption times than its alternative.

As caching is performed at the network layer, graph-based metrics may be used for deciding the node to which caching should take place. Chai et al. [7] have proposed an on-path caching algorithm based on the metric of *Betweeness Centrality (BC)*; BC metric represents the number of times that each node n lies at the sets of shortest paths, between all pairs of nodes in the graph, besides n. According to the proposed algorithm, caching should be performed only at the nodes holding the highest BC value. Chai et al. have proven that the BC algorithm provides better server hit rates and hop count rates than the CE^2 and *UniCache* algorithms.

Rossi et al. [26] have examined the suitability of a number of graph-based algorithms, *Degree Centrality (DC)*, *Betweenness Centrality (BC)*, *Closeness Centrality (CC)*, *Graph Centrality (GC)*, *Eccentricity Centrality (EC)* and *Stress Centrality(SC)*, for determining the size of the caches, e.g. proportional to the centrality value of each node rather than determing the nodes for caching the content. Based on their evaluation results, Rossi et al. have conclude that DC, which indicates the number of edges of each node n, is the most effective graph-related metric compared to the alternatives in terms of server hit rates and hop count rates.

On-path caching approaches may be further categorized based on the information used for the caching decision, i.e. *autonomous caching*; using local information, *centralized caching*; using centralized information and *dependent caching*; using information regarding other nodes in a non-centralized manner. To ease comparison, Table 1 summarizes the proposed on-path caching approaches, the approaches against which they have been compared, the metrics and topologies under which they have been evaluated and the evaluation results. In this table, symbol "-" indicates that no further information has been provided regarding this category while symbol "$a > b$" indicates that approach a results in a better performance than approach b.

Based on Table 1, three approaches, the *FIX(0.90)*, DC and *ProbCache*, seem to outperform the rest of the

Table 1: Taxonomy of the proposed on-path caching algorithms.

Proposed Technique	Comparison Technique	Caching model	Evaluation Metrics	Comparison Results	Topology type
BC [7]	UniCache, CE^2	centralized	server hit rates, hop count rates	BC > CE^2> UniCache	CAIDA (6804 nodes)
CE^2[13]	-	autonomous	-	-	-
DC, BC, CC, GC, EC, SC [26]	DC, BC, CC, GC, EC, SC	centralized	cache hit rates, hop count rates	DC > SC, BC, CC, GC, EC	Rocketfuel (up to 68 nodes)
FIX(p) [2]	CE^2	autonomous	cache hit rates, latency	CE^2> FIX(0.5) > FIX(0.3) > FIX(0.25) > FIX(0.125)	8-nodes string
LCD [25]	CE^2, FIX(p)	autonomous	cache hit rates, hop count rates, load fairness, cache diversity	FIX(0.90) > FIX(0.75) > CE^2> LCD	Rocketfuel (up to 68 nodes)
LeafNode [29]	CE^2	autonomous	hop count rates, absorption time	CE^2>LeafNode	up to 4-level binary tree
ProbCache [23]	CE^2, LCD, FIX(p)	dependent	server hit rates, hop count rates, eviction rates	ProbCache>LCD, FIX(0.70) & FIX(0.30) > CE^2	6-level binary tree

caching mechanisms. Each of these approaches follows a different caching model, *autonomous, centralized* and *dependent*, respectively. One of the main contributions of this work is the evaluation of these algorithms against each other. Based on this comparison and the previous ones performed by the research community, we expect to conclude to the nature of the caching system that would be more beneficial for an ICN architecture.

3. PROBABILISTIC-PD ALGORITHM

Based, on the information summarized in section 2, one can observe the absence of the content popularity as a criterion to the caching decision. Content popularity is an important factor, able to affect the performance of a caching algorithm and the network as a whole [5, 25]. Therefore, it should be taken into account. Content popularity has been applied on a number of cache replacement policies and replication algorithms defined in the areas of web proxies and CDNs, e.g. [14, 16, 22].

Measurement studies in web caching have highlighted the problem of *cache pollution* due to *one-timer* objects [22]. One-timer objects is a term used to identify objects that are requested only once while cache pollution is a term used to identify the case where one-timer objects are cached. Cache pollution prevents the caching of popular objects, resulting into higher cache miss rates and higher network traffic rates. According to the same studies, one-timer objects correspond into approximately 45-75% of the total amount of requests. As on-path caching is expected to serve a much higher catalog size than object replication mechanisms, under more severe restrictions, such as the cache capacity restrictions [2, 19], the prevention of the cache pollution problem becomes an even more important prerequisity.

Towards this direction we propose a probabilistic on-path caching algorithm that takes into account the popularity of the content. Inspired by the *Local Greedy* algorithm, proposed by Kangasharju et al. [16] for content replication on CDNs, we propose a probabilistic algorithm, called *Prob-PD*. The Prob-PD algorithm consists of two factors, the content's popularity ratio P, observed on a node and the distance ratio between the same node and the source serving the content, D.

The idea behind popularity-based caching is that more popular contents will satisfy a higher portion of the total requests. Therefore, caching popular contents should be preferred. At this point one should make a decision regarding the way that content popularity is caclulated and define the behavior set against both popular and unpopular contents. Based on this criterion, content popularity may be divided into *static-content popularity* and *dynamic-content popularity*, respectively.

In a static-content popularity approach, contents are distinguished from each other using a threshold h. Contents with a number of requests higher than h are considered to be popular while contents with a number of requests lower than h are considered to be unpopular [8, 15, 21, 24]. Unpopular contents are excluded from the caching decision. Static-content popularity approaches require the definition of a proper threshold. Due to the volatile nature of ICN architectures, we expect such a definition to be quite challenging. As such, static-

Table 2: Parameters of the system model used for evaluation.

Parameter	Symbol	Value	Definition
No. of nodes	|N|	97	Total No. of nodes
No. of backbones	|B|	39	Total No. of backbone nodes
No. of gateways	|G|	58	Total No. of gateway nodes
Capacity of links	BW	40GB	Available bandwidth
Catalog Size	|O|	1000,10000	Total No. of objects
Object Size	o_i	$\forall o_i, i\epsilon\|O\| \sim N(10000KB, 9800KB)$	Size of object o_i in KB
Chunk Size	Ch	$10KB$	Chunk size in KB
Contents Size	$\|C\|$	$\sum_{i=1}^{\|O\|} o_i/Ch$	Total No. of chunks
Cache Size	cs_i	$\forall cs_i, i\epsilon\|N\|, cs_i\epsilon\{1, 10, 100, 1000\}$	Cache capacity of node i in chunks
Consumers Size	u_i	$\forall u_i, i\epsilon\|G\| \sim U(100, 300)$	No. of users on gateway i
Rank Parameter	q	$q\epsilon\{0.5, 5\}$	Rank parameter of the Z-M object popularity distribution
Zipf Exponent	α	$\alpha\epsilon\{0.8, 1.0, 1.5, 2.0, 2.5\}$	Exponent of the Z-M object popularity distribution
Arrival rate	λ	1.0	Exponential request arrival rate
Control window	W	∞	No. of requests able to sent with no reply

content popularity approaches usually result into out of date calculations and unutilized cache capacity [15].

Dynamic-content popularity on the other hand is defined as the number of requests observed during a time interval Δ_t [18, 27]. Consequently, the popularity of a content is concluded by comparing its request rates against each other's. A common technique to provide an up to date content popularity pattern is to sort the contents in a decreasing order [4, 16], similarly to a LFU replacement policy. Therefore, we believe that applying a dynamic-content popularity approach on an ICN architecture instead of a static one would be more beneficial. However, one should not neglect the disadvantage deriving from the same feature, i.e. the constant comparison of the request rates. Towards this direction, we propose a dynamic-content popularity approach that reduces the number of comparisons to a minimum.

We have already stated the reasons why we chose popularity to be one of the factors that construct our caching algorithm. We now attempt to explain the reasons why we chose distance to be the combined factor. Similarly to any other caching technique, on-path caching is requested to answer the question of where to cache a content. We slightly change this question into whether a content should be cached on a node based on a network-related criterion. Latency reduction is one of the most preferable goals with regard to network performance, therefore one of the most known network metrics. The number of hops has been defined as a good estimation of the latency metric, being used in routing protocols and CDN replication algorithms [16, 20].

In order to explain the caching algorithm further, we first define some notations that we summarize in Table 2. For the rest of the section, let i denote the node performing the caching decision and j denote the content on which the caching decision is applied. Let $r_{i,j}$ denote the number of requests recorded on node i for content j, with $\sum_{j=1}^{J\leqslant Ch} r_{i,j}$ being the total number of requests seen and $d(i, i')$ denote the distance between nodes i and i', using the number of hops as a metric. We then define the $Prob - PD_{i,j}$ algorithm as follows:

$$Prob - PD = \underbrace{\frac{r_{i,j}/\Delta_t}{\sum_{j=1}^{J\leqslant|C|} r_{i,j}/\Delta_t}}_{P} \times \underbrace{\frac{d(i, src)}{d(dst, src)}}_{D} \quad (1)$$

where, P represents the dynamic popularity calculation of a content j, constructed by the number of requests recorded for content j, during a time interval Δ_t, divided by the total number of requests recorded during the same time interval, on a node i. In order not to introduce any additional overhead to the infrastructure of a node, we define Δ_t to be the time between the arrival of the first request regarding content j and the satisfaction of it. That way, each content is limited to one comparison only against the rest of the contents, minimizing the complexity and overhead that dynamic popularity calculations apply. The D factor, is constructed based on the distance between node n and the source *src* serving the request, normalized by the distance between *src* and the node requesting the content *dst*. The D factor represents the benefit of caching the content on the current node against the cost of retrieving the content from the original source. Our goal is to examine how beneficial may be the combination of these two factors regarding the ICN on-path caching problem.

4. SYSTEM MODEL

In this section we provide a thorough analysis of the evaluation system model. The evaluation is based on the *ndnSIM* simulator, an ns-3 module that adopts the *Named Data Networking (NDN)* communication model [13]. A summary of the model can be found in Table 2.

In order to ease readers to relate the results with those presented in other publications, a real network topology, Exodus AS-3967 is used, based on the Rocketfuel traces [30]. The topology, shown in Fig.1, consists of 94 nodes, i.e. 39 backbones and 58 gateways. Each node is equipped with a NDN stack. Consumer and producer applications can only be installed on a gateway node. In particular, one producer is assumed for each evaluation scenario. The selection of a gateway for the producer installation is based on the metric of connectivity degree; a node with connectivity degree equal to 5 is chosen, where 1 is the minimum and 14 is the maximum.

As an attempt to provide realistic evaluation scenarios, a simulation scenario based on the YouTube traffic is determined. However, as the exact simulation of such a scenario is computationally expensive [25], the normalization of some characteristics is necessary so as to adopt the model; we decrease the catalog size |O| from 10^8 [6, 35] to 10^4 and the content store (CS) size, cs_i from 10GB [2] to 1MB. It is worth mentioning that the ratio between the initial values and the ones concluded remains the same. Object sizes follow a normal distribution of mean 10MB and standard deviation 9.8MB [11]. In order to be able to study the effect of the popularity factor on the performance of the algorithms, a Zipf-Mandelbrot (Z-M) object popularity distribution of $\alpha\epsilon\{0.8, 1.0, 1.5, 2.0, 2.5\}$ and $q\epsilon\{0.5, 5\}$ [6, 34, 12] is defined. Contents in CS are replaced using the Least Recently Used (LRU) policy [13, 25].

Finallizing the system model, a mean number of 200 consumers is installed on each gateway, following a uniform distribution. A consumer generates object requests. Each object request corresponds to a sequence of chunk requests, equal to the size of the object divided by the chunk size, 10KB [11, 35]. Request arrivals follow an exponential distribution of λ equal to 1.0.

5. EVALUATION RESULTS

Using the system model desrcibed in section 4 and the evaluation metrics of cache hit rates, cache replacement rates, delivery times and hop count rates, a report of the evaluation results of the *CE*^{2}*,DC,FIX(0.90),ProbCache* and *Prob-PD* algorithms is provided. Each evaluation result corresponds to the mean value of 10 simulation runs under a set of parameters α, q and cs_i; due to space limitations and in order to ease readability, only the mean values of the evaluation results are displayed. In order to be able to compare the caching mechanisms between each other an average value derived from the summation of each evaluation result with respect to each parameter is used. Again, due to space restrictions, a modification of the names of four of the algorithms is applied, i.e. *CE,P90,PC* and *PD* for the *CE*^{2}*, FIX(0.90), ProbCache* and *Prob-PD* algorithms, respectively.

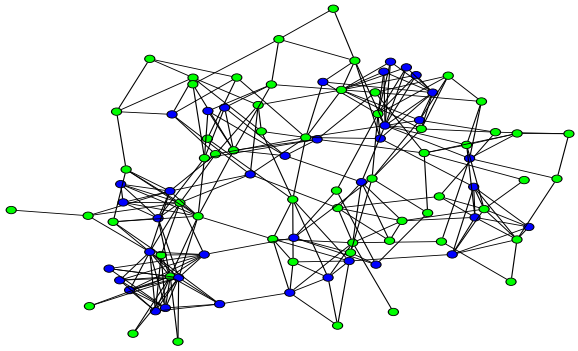

Figure 1: Exodus Topology (AS-3967)

Fig.2 illustrates the performance of the caching mechanisms with regard to the cache hit rates for a range of parameters, i.e. $\alpha\epsilon\{0.8, 1.0, 1.5, 2.0, 2.5\}, q\epsilon\{0.5, 5\}$ and $cs_i\epsilon\{10, 100, 1000\}$. According to Fig.2, the cache hit rates can be highly affected by the parameter α; the number of cache hits decreases as α increases from 0.8 to 1.0 and increases again when $\alpha \geq 1.0$. An exception to this behavior constitute the CE and P90 algorithms for which the cache hit rates decrease when $0.8 \leq \alpha \leq 1.5$.

In a Zipf-Mandelbrot distribution, the parameters α and q determine the probability of an object to be requested. As α and q increase, the requests get limited to a more strict subset of objects. The behavior of the algorithms suggests that when $0.8 \leq \alpha \leq 1.0$ the pattern of requests is neither too scarce nor too concentrated to cause a cache hit; a scarce pattern of object requests may as well correspond to an increase on the cache hits given the number of consumers u_i on each gateway.

Towards analyzing Fig.2, two approaches, the PC and PD seem to outperform the rest of the alternatives, with the PC approach performing slightly better than the PD; approximately 1.7 cache hits on average. As expected, the CE and P90 approaches result in an almost identical performance of 0.6 cache hits difference in favor of the P90. The DC approach provides the lowest rates on cache hits, i.e. about 31.5 less than the PC.

An important point that we owe to highlight is the lack of the PD algorithm to provide a comparable performance with regard to the alternatives. The explanation for this outcome lies on the calculation of the popularity factor P. As a content competes against the sum of the contents requested during the time interval Δ_t, only contents with high popularity rates are cached. Consequently, the algorithm misses the cache hits derived from the less popular contents. Of course, the behavior of the algorithm is also related to the traffic

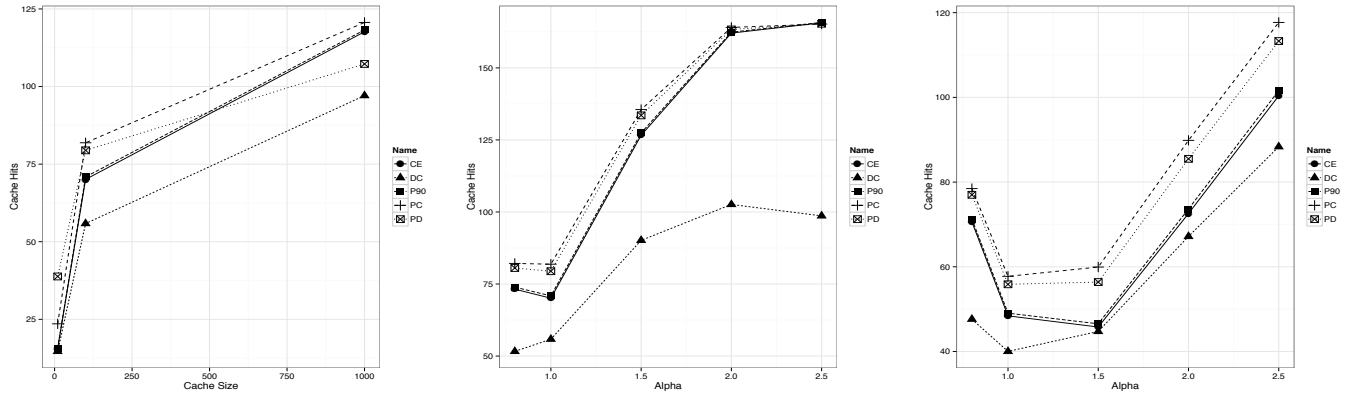

Figure 2: Average cache hit rates for |O|=10.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{10, 100, 1000\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$ (right).

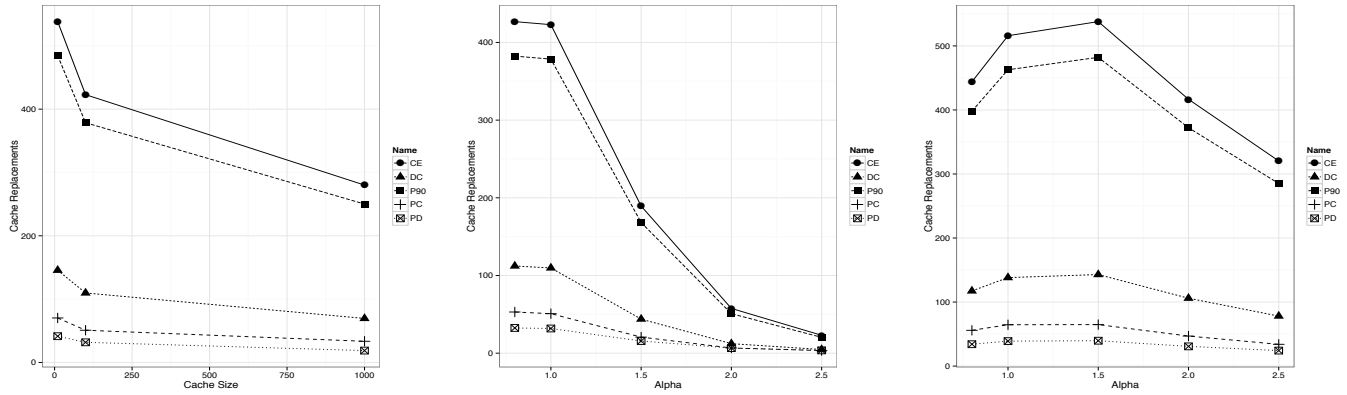

Figure 3: Average replacement rates for |O|=10.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{10, 100, 1000\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$ (right).

model adopted. To this end, a second system model, explained later in this section is also evaluated.

In order to verify our claims we plot the replacement rates, Fig.3, of each approach with regard to the same parameters. If our assumption about the PD algorithm is true, the cache replacement rates should be relatively low. Indeed, Fig.3 indicates that the PD algorithm corresponds to the lowest cache replacement rates while the PC and DC approaches correspond to an average increase of 15.5 and 63, respectively. As expected, the CE and P90 approaches produce the highest cache replacement rates, i.e. 300 more than the PD algorithm.

Finalizing the presentation of the evaluation results for the specific system model we also plot the delivery times and hop count rates of each approach, Fig.4 and Fig.5. As the number of hops is assumed to be a good estimation of the latency metric, both figures conclude to a similar outcome. Somewhat suprisingly to Fig.2, the DC algorithm seems to correspond to a lower evaluation metric compared to the PC and PD alternatives. The difference between the approaches is estimated at 0.002 and 0.003 for the delivery times and 0.021 and 0.18 for the hop count rates. The result suggests that caching and network evaluation metrics do not strictly align with each other. As the final goal of a caching algorithm is the reduction of the latency and the reduction of the network traffic, both of which can be estimated based on the delivery times and the hop count rates, the outcome suggests that a DC approach may as well correspond to high benefits. Moreover, the outcome comes in contrast to previous works where CE has been identified as a fast delivery algorithm [7].

In order to explore whether the performance of our proposal is indeed affected by the traffic model, the alteration of two main system parameters is introduced, mainly, the reduction of the catalog size $|O|$ from 10.000 to 1.000 and the reduction of the CS size from 100 to 10. Similar to the system model described in section 4, the ratio of the two parameters remains the same.

Fig.6 presents the cache hit rates of the new system model with respect to the parameters α, q and cs_i. According to Fig.6, our proposal, PD corresponds to the highest cache hit rates with a minimum difference of 5.5, against the PC approach and a maximum difference of 18.2, against the DC approach. Similar to Fig.2, the variance of cache hits between the CE and P90 ap-

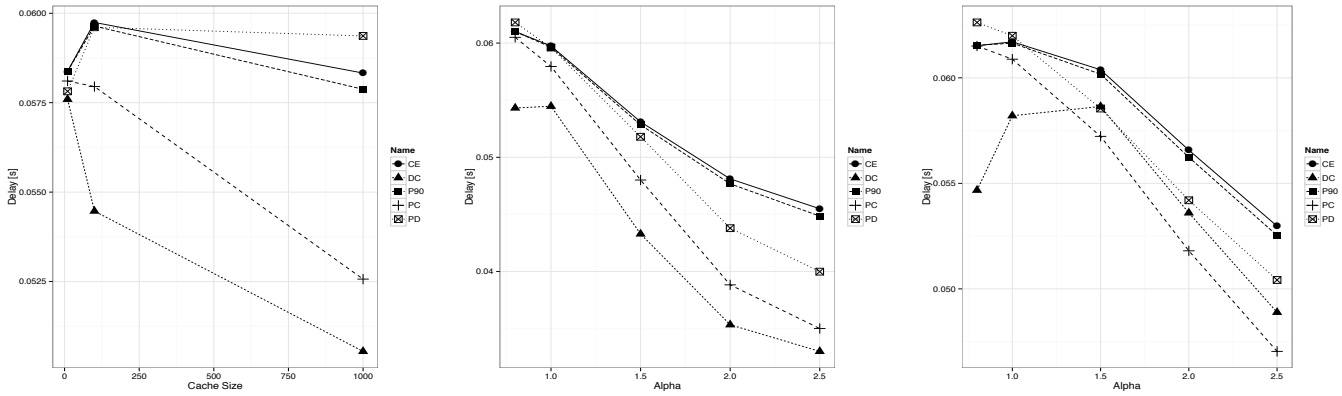

Figure 4: Average delay times for |O|=10.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{10, 100, 1000\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$ (right).

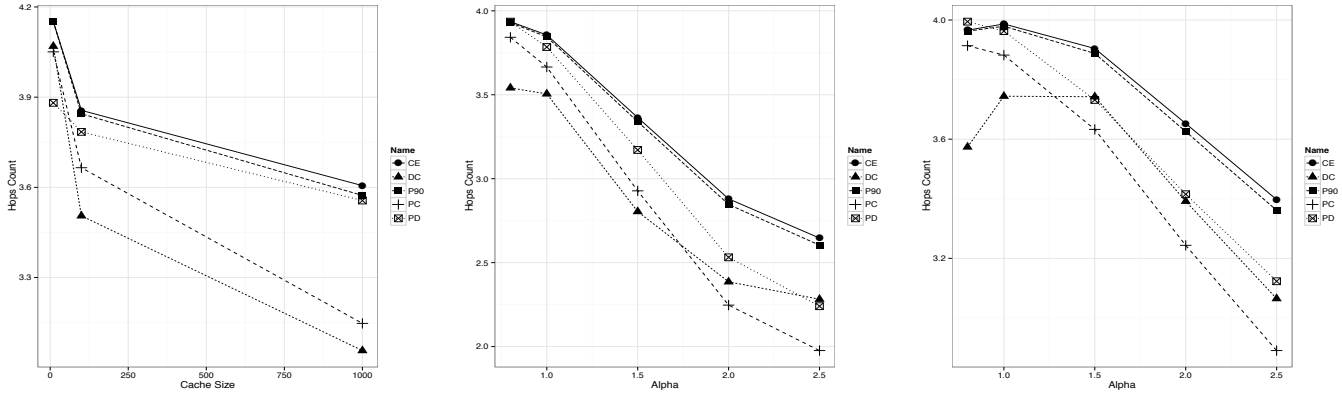

Figure 5: Average hop count rates for |O|=10.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{10, 100, 1000\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$ (right).

proaches is negligible. It is worth noting that in contrast to the previous evaluation model, the cache hit rates are no longer decreased as the parameter α increases. The main reason for this is a more clear pattern of requests.

Fig.7 presents the cache replacement rates with regard to the same parameters. Based on the aforementioned results and the ones reported in Fig.3, one can easily observe that the alteration of the system model has no significant impact on the trend of the algorithms.

In contrast to the cache replacement rates, the plots of both the delivery times, Fig.8 and the hop count rates, Fig.9 are no similar to the ones recorded in the previous evaluation. More precisely, Fig.8 suggests that the variation of the delivery times of the algorithms is as low as 0.0001, i.e. with the lowest value being 0.0571 for the PD algorithm and the highest value being 0.0581 for the CE algorithm. The reported variation is lower than the one recorded in Fig. 4. Fig.9 concludes to an average value of 3.85 and 4.05 hop counts for the PD and CE approaches, respectively. The difference between the PD algorithm and the best alternative, PC is as low as 0.06, which is higher than the one recorded in Fig. 5 between the DC and the PC algorithms.

Based on the aforementioned results, one can conclude that the performance of an on-path caching algorithm can be affected from a range of parameters, the most important of which is perhaps the catalog size |O|. The catalog size is able to significantly affect the traffic model of the system and define a more scarse or a more frequent pattern of requests. Analyzing the impact of the parameter |O| a bit further, we shortly recall the dependence of the DC and PC approaches with regard to the delivery times and the hop count rates. More precisely, for |O|=10.000, the PC approach outperforms the DC approach while for |O|=1.000, the PC approach outperforms the DC approach. The pattern of requests is even more critical to the performance of our algorithm due to the calculation of the popularity factor P, explained earlier in this section. Towards this direction, alternative options for the calculation of the popularity factor are concidered. A potential alternative is the modification of the time interval Δt to a potentially larger fixed time interval. The main reason for this is that a larger time interval would probably result into a better categorization of the popularity of the contents. We attempt to fulfill this as a future work.

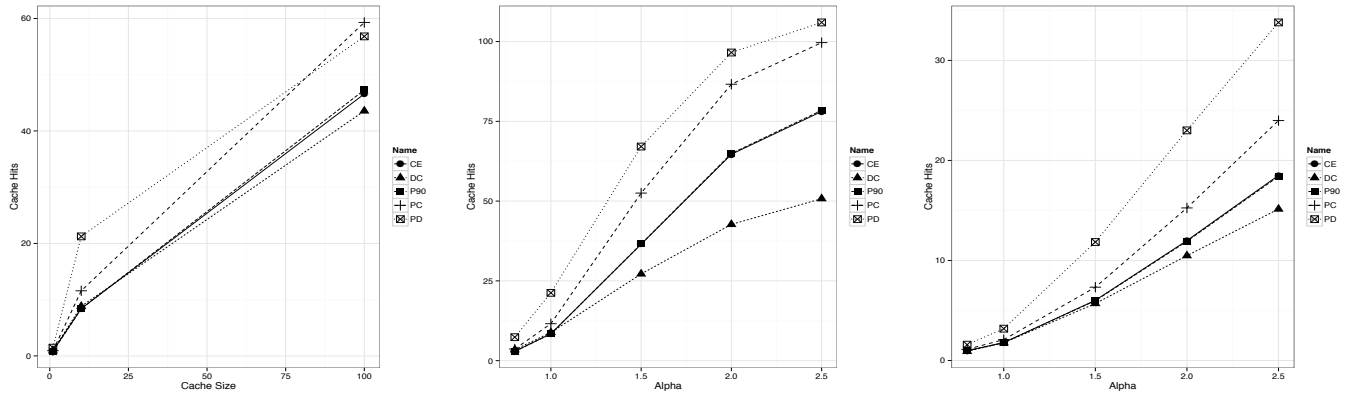

Figure 6: Average cache hit rates for |O|=1.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{1, 10, 100\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$ (right).

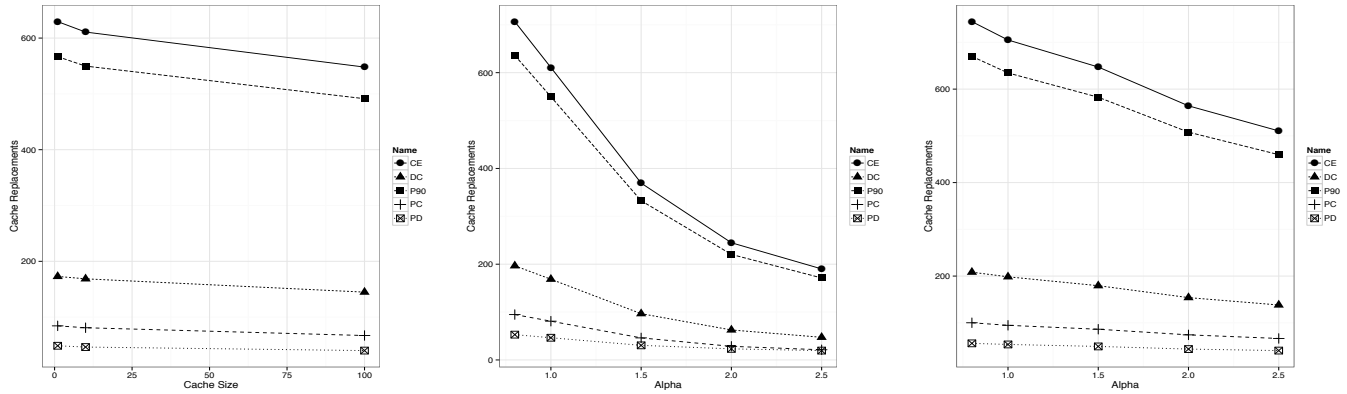

Figure 7: Average replacement rates for |O|=1.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{1, 10, 100\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$ (right).

6. CONCLUSIONS

In this paper, we have described the existing on-path caching algorithms for the ICN architectural model and categorized them against their properties. We have further proposed a probabilistic caching algorithm, *Prob-PD*, to enhance performance which we evaluated against the alternatives via simulations. Our results indicate that the performance of an on-path caching algorithm may be considerably affected by the catalog size |O|. More precisely, our approach may provide significant gains if certain conditions are met, such as $|O| \leq 10.000$. The explanation for this outcome lies on the calculation of the popularity factor P. Towards solving this dependency we intend to explore alternatives that would allow the algorithm to perform in a wider scale.

7. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[2] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop of the ACM CoNEXT Conference*, number 5, 2010.

[3] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications (SIGCOMM '04), Portland, OR, USA, August 2004*, pages 343–352.

[4] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. In *Proceedings of the 29th IEEE Conference on Computer Communication (INFOCOM '10), San Diego, CA, USA, March 2010*, pages 1–9.

[5] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling data transfer in content-centric networking. In *Proceedings of the 23rd International Teletraffic Congress (ITC'11), San Francisco, CA, USA, September*, pages 111–118, 2011.

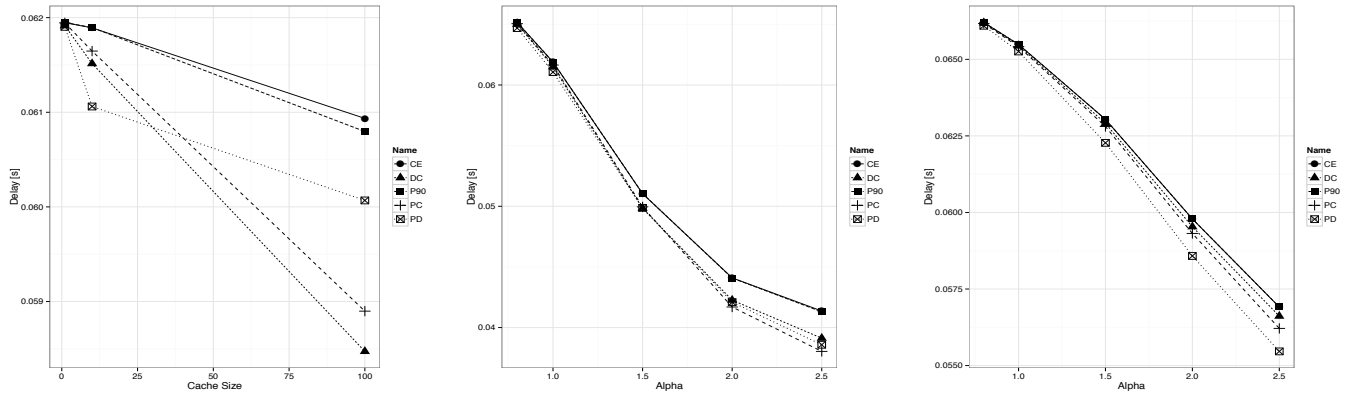

Figure 8: Average delivery times for |O|=1.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{1, 10, 100\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$ (right).

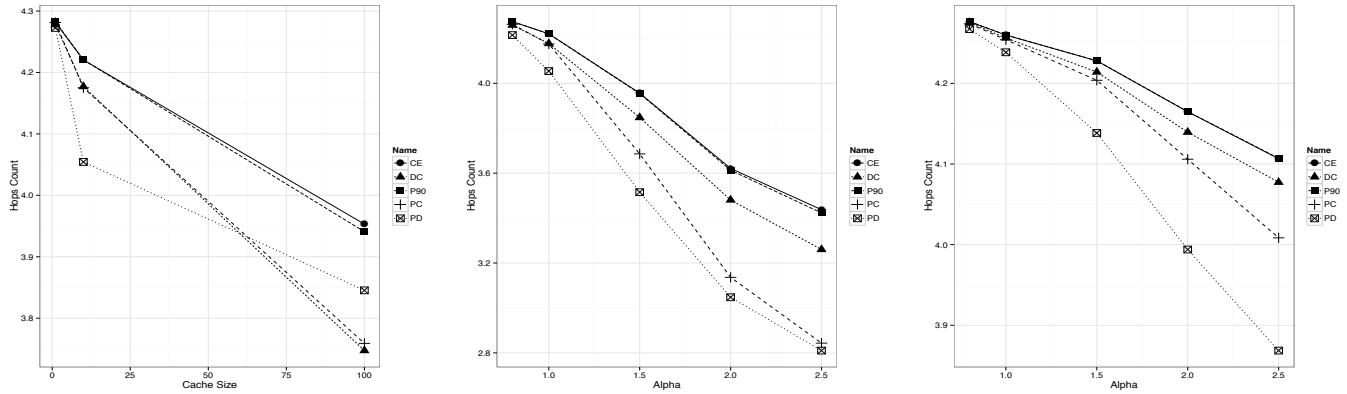

Figure 9: Average hop count rates for |O|=1.000: $(a.)\, a = 1.0, q = 0.5, cs_i \epsilon \{1, 10, 100\}$ (left), $(b.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$ (middle), $(c.)\, \alpha \epsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$ (right).

[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC '07), San Diego, CA, USA, October 2007*, pages 1–14, 2007.

[7] W. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks. In *Proceedings of the 11th International IFIP TC 6 Conference on Networking, Brooklyn, NY, USA, May 2012*, pages 27–40.

[8] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305–1314, 2002.

[9] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *Proceedings of the 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN '12), Orlando, FL, USA, March 2012*, pages 316–321.

[10] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[11] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM Conference on Internet Measurement (IMC '07), San Diego, CA, USA, October 2007*, pages 15–28.

[12] M. Hefeeda and O. Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Transactions on Networking (TON)*, 16(6):1447–1460, 2008.

[13] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09), Rome, Italy, December 2009*, pages 1–12.

[14] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the internet. In *Proceedings of the 20th IEEE Conference on Computer Communication (INFOCOM '01),*

Anchorage, AK, USA, April 2001, pages 31–40.

[15] T. Janaszka, D. Bursztynowski, and M. Dzida. On popularity-based load balancing in content networks. In *Proceedings of the 24th International Teletraffic Congress (ITC'12), Krakow, Poland, September 2012*, pages 1–8.

[16] J. Kangasharju, J. Roberts, and K. Ross. Object replication strategies in content distribution networks. *Elsevier Journal on Computer Communications*, 25(4):376–383, 2002.

[17] N. Laoutaris, H. Che, and I. Stavrakakis. The lcd interconnection of lru caches and its analysis. *Elsevier Journal on Performance Evaluation*, 63(7):609–634, 2006.

[18] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed selfish replication. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1401–1413, 2006.

[19] U. Lee, I. Rimac, and V. Hilt. Greening the internet with content-centric networking. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy '10), Passau, Germany, April 2010*, pages 179–182.

[20] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *Proceedings of the 8th IEEE Conference on Computer Communication (INFOCOM '99), New York, NY, USA, March 1999*, pages 1282–1290.

[21] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong. Popularity-driven coordinated caching in named data networking. In *Proceedings of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '12), Austin, TX, USA, October 2012*, pages 15–26.

[22] A. Mahanti, D. Eager, and C. Williamson. Temporal locality and its impact on web proxy cache performance. *Elsevier Journal on Performance Evaluation*, 42(2):187–203, 2000.

[23] I. Psaras, W. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the 2nd Workshop on Information-Centric Networking, Orlando, FL, USA, March 2012*, pages 55–60.

[24] M. Rabinovich, I. Rabinovich, R. Rajaraman, and A. Aggarwal. A dynamic object replication and migration protocol for an internet hosting service. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS '99), Austin, TX, USA, May 1999*, pages 101–113.

[25] D. Rossi and G. Rossini. Caching performance of content centric networks under multi-path routing (and more). Technical report, Telecom ParisTech Ecole, November 2011.

[26] G. Rossini and D. Rossi. On sizing ccn content stores by exploiting topological information. In *Proceedings of the 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN '12), Orlando, FL, USA, March 2012*, pages 280–285.

[27] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas. Autonomic cache management in information-centric networks. In *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '12), Maui, HI, USA, April 2012*, pages 121–129.

[28] V. Sourlas, P. Flegkas, G. Paschos, D. Katsaros, and L. Tassiulas. Storage planning and replica assignment in content-centric publish/subscribe networks. *Elsevier Journal on Computer Networks*, 55(18):4021–4032, 2011.

[29] V. Sourlas, G. Paschos, P. Flegkas, and L. Tassiulas. Caching in content-based publish/subscribe systems. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09), Honolulu, HI, USA, November 2009*, pages 1–6.

[30] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking (TON)*, 32(4):133–145, 2002.

[31] A. Vakali and G. Pallis. Content delivery networks: Status and trends. *IEEE Journal on Internet Computing*, 7(6):68–74, 2003.

[32] M. Walfisha, H. Balakrishnana, and S. Shenkerb. Untangling the web from dns. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, USA, March 2004.*

[33] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys and Tutorials*, (1–26), 2013.

[34] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of the 1st ACM European Conference on Computer Systems (EUROSYS '06), Leuven, Belgium, April 2006*, volume 40, pages 333–344.

[35] J. Zhou, Y. Li, V. K. Adhikari, and Z.-L. Zhang. Counting youtube videos via random prefix sampling. In *Proceedings of the 7th ACM Conference on Internet Measurement (IMC '07), Berlin, Germany, November 2011*, pages 371–380.