Information-Centric Networking: A Thorough Evaluation of Popularity-based Probabilistic On-path Caching

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

Abstract

The original Internet architecture was conceived to establish a connection between two participants. However, Internet usage today is dominated by content distribution and retrieval that comes in contrast to the host-based communication model of the network infrastructure.

Information-Centric Networking (ICN) provides an alternative to the traditional Internet architecture by focusing on content. ICN networking is based on the publish-subscribe paradigm and the features of naming and in-network caching. In-network caching can be categorized into *off-path caching* and *on-path caching* with regard to the location of caches and the delivery path.

The contribution of this paper lies on the placement of copies in on-path in-network caching. Our aim is to examine the suitability of a probabilistic algorithm, *Prob-PD*, based on two variables, the contents popularity rates P and the distance ratio of each node from the source D, using the performance metrics of cache hit rates, cache eviction rates, hop counts and content delivery times.

To this end, we present a thorough evaluation of the proposed caching mechanism and published alternatives based on Rocketfuel traces and YouTube traffic with regard to a number of parameters such as the catalog size $|O|$ and the chunk size Ch. Our results suggest that the performance of the algorithms may be considerably affected by both these factors. In particular, our approach may provide significant gains if certain conditions are met, such as $|O| \leq 10.000$ or $Ch \leq 10$KB.

1 Introduction

The original design of protocols for the Internet was provided for information exchange between two participants. However, usage patterns and technologies have changed since these protocols were developed and todays Internet usage is dominated by the distribution and retrieval of content. According to a recent technical paper by CISCO [12], the annual traffic of the Internet is expected to exceed the size of 1.4 zettabytes by the end of 2017 with almost 80% of the traffic being video. This mismatch between the original design and the current use of the network infrastructure results in a number of disadvantages and difficulties with regards to availability, mobility, multihoming, scalability and performance [3, 14].

Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) networks are the first steps towards a content-oriented Internet architecture. However, both technologies operate at the application layer and depend on the underlying traditional network infrastructure of the transfer of information between two endpoints [22, 25, 34]. This mismatch of higher-layer and lower-layer components may result in reduced performance [9, 24].

Information-Centric Networking (ICN), an alternative to the traditional Internet architecture, focuses on information dissemination and information retrieval. ICN networking is based on the publish-subscribe paradigm where 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 consumers request content from a content notification service by subscribing to it.

Content publication and retrieval are accomplished via *content identifiers*. ICN architectures identify content resources such as services, web pages, songs etc. or parts of a content resource, chunks or packets, using a content identifier. Content identifiers should involve no information that would bind the content to a specific location. If this constraint is met, content can be freely replicated and cached in the network infrastructure. Approaches to caching can be categorized into *off-path caching* and *on-path caching* with regard to the location of caches.

Off-path caching, also referred as content replication, aims to replicate content within a network regardless of the forwarding path. Off-path caching is usually centralized. Its operation is based on the selection of content to be replicated and the selection of nodes to be hosting

Table 1: Evaluation metrics used in the literature review.

Evaluation Metric	Metric Relation	Definition
Hop count	Network-based	No. of nodes a content request has travelled before been satisfied.
Hop count rate	Network-based	Hop count to the no. of hops towards a content source.
Delivery time	Network-based	Total time experienced to retrieve content.
Server hit rate	Cache-based	No. of content requests satisfied by a server to the no. of content requests expressed in a network.
Cache hit rate	Cache-based	No. of content requests satisfied by a node to the no. of content requests expressed in a network.
Cache eviction rate	Cache-based	No. of replacements occur on a node's cache memory when is full.
Absorption time	Cache-based	The time for which content remains cached in a node's cache memory.

the replicas, concluded via network monitoring and load balancing mechanisms. Content replicas are finally advertised to a content notification service. The off-path caching problem is equivalent to the CDN content replication and the web cache placement problems [31, 35].

On-path caching is based on the storage capacity of the infrastructure; routers in ICN are equipped with cache memory and enabled with a caching capability. Thus, on-path caching is accomplished at the network layer of the infrastructure, along the delivery path(s), which in turn impedes a number of advantages and challenges, e.g. caching is independent of the application, caching decision is limited to the content propagated along the delivery path(s) and to the nodes lying on the delivery path(s), 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. In addition to this, on-path caching does not follow any structural model; the topology is considered arbitrary.

The focus of this paper lies on the efficiency of on-path caching mechanisms. Towards this goal, we propose a probabilistic algorithm, Prob-PD, based on two variables, the contents 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 the paper is structured as follows. Section 2 summarizes the related work of on-path caching. Section 3 introduces the Prob-PD algorithm. Section 4 describes the evaluation system model. Section 5 presents the evaluation results of the algorithm against the alternatives. Section 6 is devoted to the conclusions. This paper constitutes a continuation of our previous initial work [13] by providing a thorough evaluation.

2 Related Work

Due to the integration of on-path caching with the forwarding procedure and the challenges deriving from it, a number of on-path caching algorithms have been proposed in the literature. In this section, a summary of the algorithms and their evaluation results is provided. A description of the evaluation metrics is available in Table 1.

A range of existing on-path caching approaches base the decision to cache a content on a node on a probability p. This probability can be either a fixed value or a dynamic value calculated via a mathematical formula. Fixed-value approaches, *FIX(p)*, may base their decisions on an arbitrary value [2, 14], e.g. $CE^2$of $p=1$ or on the number of nodes in a delivery path, e.g. UniCache [8, 6]. These simplistic approaches neglect the variable nature of content requests and network topologies, which in turn results in lower performance compared to the alternatives, with regard to the cache hit rates, hop count rates and content delivery times [26, 28, 6].

In order to address this, *ProbCache* [26] uses a dynamic probabilistic algorithm that defines the number of replicas to be cached along the delivery path based on the cache memory of the nodes lying between the node deciding the caching and the consumer. This approach has the advantage to be more reactive than fixed approaches, thus, exhibit lower server hit rates, hop count rates and cache eviction rates compared to the FIX(p), $CE^2$and LCD; LCD caches a copy of the requested content one hop closer to the client each time a content request arrives [31]. LCD has been also shown to be outperformed by FIX(p) algorithms, in terms of cache hit rates [28].

LeafNode [32] is an other algorithm proposed for the on-path caching problem that caches content at the last node of the delivery path. LeafNode targets to keep content as close to the consumers as possible. An evaluation of this approach against the CE^2has shown that it provides higher hop count rates and lower absorption times.

Due to the integration of on-path caching to the network layer of the architecture, graph metrics may be used for deciding the node(s) to perform caching. Graph-based approaches such as *Betweeness Centrality (BC)* [6] and *Degree Centrality (DC)* [29] react to the topology of the network by taking into account the nodes on the de-

Table 2: Taxonomy of the proposed on-path caching algorithms.

Proposed Technique	Comparison Technique	Caching model	Evaluation Metrics	Comparison Results	Topology type
BC [6]	UniCache, CE^2	centralized	server hit rate, hop count rate	BC > CE^2> UniCache	CAIDA (6804 nodes)
CE^2[14]	-	autonomous	-	-	-
DC, BC, CC, GC, EC, SC [29]	DC, BC, CC, GC, EC, SC	centralized	cache hit rate, hop count rate	DC > SC, BC, CC, GC, EC	Rocketfuel (up to 68 nodes)
FIX(p) [2]	CE^2	autonomous	cache hit rate, delivery time	CE^2> FIX(0.5) > FIX(0.3) > FIX(0.25) > FIX(0.125)	8-nodes string
LCD [28]	CE^2, FIX(p)	autonomous	cache hit rate	FIX(0.9) > FIX(0.75) > CE^2> LCD	Rocketfuel (up to 68 nodes)
LeafNode [32]	CE^2	autonomous	hop count rate, absorption time	CE^2>LeafNode	up to 4-level binary tree
ProbCache [26]	CE^2, LCD, FIX(p)	dependent	server hit rates, hop count rate, cache eviction rate	ProbCache>LCD, FIX(0.7) & FIX(0.3) > CE^2	6-level binary tree

livery paths. BC takes into account the number of times a node is included in the shortest paths between sources and consumers while DC is based on the number of connections of a node. These approaches may lead to advantages when requests are well distributed but they neglect the frequency and distribution of their occurrence. In addition, graph-based approaches are based on the collection of centralized information that comes in contrast to the distributed nature of ICN networking. DC constitutes an exception to this rule. DC has been concluded as the most effective graph metric in terms of server hit rates and hop count rates while BC has been concluded to be the most effective approach compared to UniCache and CE^2, with regard to the same evaluation metrics.

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 non-centralized information regarding other nodes. To ease comparison, Table 2 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 for this category while symbol *"$a > b$"* indicates that approach a outperforms approach b.

Based on Table 2, the approaches FIX(0.9), DC and ProbCache outperform the rest of the caching mechanisms. Each approach follows a different caching model, autonomous, centralized and dependent, respectively. One of the contributions of this paper 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 caching system that would be more beneficial for an ICN architecture.

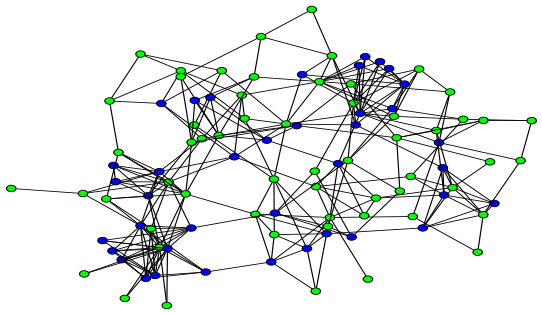

Figure 1: Exodus Topology (AS-3967)

3 Probabilistic-PD Algorithm

Based on on-path caching literature, one can observe the absence of content popularity to the caching decision. Content popularity has been concluded to significantly affect the performance of caching algorithms [4, 28]. The idea behind content popularity is that popular contents will satisfy a higher portion of requests. Therefore, it should be taken into account. Content popularity has been used in cache eviction policies and content replication mechanisms [15, 17, 23].

Measurement studies in web caching have highlighted the problem of *cache pollution* due to *one-timer objects* [23]. One-timer objects are objects requested only once while cache pollution is the case where one-timer objects are cached. Cache pollution results in higher cache miss rates and network traffic. According to the same studies, one-timer objects correspond to 45-75%

of the content requests. As on-path caching is expected to serve a much higher number of objects than replication mechanisms, under more severe restrictions such as cache availability [2, 19], cache pollution prevention becomes an important prerequisite. Therefore, caching popular contents should be preferred. Based on the way content popularity is calculated, a *static-content popularity* or a *dynamic-content popularity* may be defined.

In a static-content popularity approach, contents are categorized using a threshold h. Contents with a number of requests higher than h are defined as popular while contents with a number of requests lower than h are defined as unpopular [7, 16, 21, 27]. Unpopular contents are excluded from the caching decision. Due to the volatile nature of ICN architectures, we expect the definition of a threshold to be challenging, resulting in out of date calculations and unutilized cache capacity [16].

In a dynamic content-popularity approach, the popularity of a content is determined by comparing its number of requests against the number of requests of the rest contents, during a time interval Δ_t [18, 30]. Thus, dynamic content-popularity approaches introduce an important computational overhead to the system. To this end, we propose a dynamic-content popularity approach that reduces the number of comparisons to a minimum.

Content popularity defines the content to be cached. However, similarly to any caching technique, on-path caching is requested to define where to cache content. As the primary goal of caching mechanisms is the reduction of delivery times, distance metrics may be used to satisfy this question. Hop counts metric has been defined to be a good estimation of the delivery times [17, 20]. Thus, hop counts represents one of the factors of our algorithm.

We propose a dynamic probabilistic on-path caching algorithm, called Prob-PD. The Prob-PD algorithm is based on two factors, the contents popularity ratio P, observed on a node and the distance ratio between the same node and the source serving the content D. Both factors are observed and calculated over time in order to react to the volatility of the network.

To explain the caching algorithm further, a number of notations are defined that are summarized in Table 3. 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 on node i for content j, with $\sum_{j=1}^{J\leqslant Ch} r_{i,j}$ being the total number of requests and $d(i,i')$ denote the distance between nodes i and i' in hop counts. We then define the $Prob-PD_{i,j}$ algorithm as follows:

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

where, P is the dynamic popularity of a content j structured by the number of requests for content j during the time interval Δ_t to the total number of requests during the same time interval on node i, thus, $P\varepsilon[0,1]$. In order not to introduce any overhead to the infrastructure of a node, we define Δ_t to be the time between the arrival of the first request for content j and the satisfaction of it. This way, a content is limited to one comparison against the rest of the contents, minimizing the complexity and overhead that dynamic popularity calculations apply. D factor is structured based on the distance between node i and the node serving the request *src*, normalized by the distance between *src* and the consumer *dst*. A content source in this case is any node that holds the content. d(i,src) and d(dst,src) calculations are based on the inclusion of two extra fields in the packet format; d(i,src) value is updated along the forwarding path, from a consumer to a content source, while d(dst,src) value is updated along the delivery path, from a content source towards a consumer. As forwarding paths and delivery paths in ICN follow the same set of nodes, $D\varepsilon[0,1]$. D factor indicates the potential gain of retrieving the content locally against the cost of retrieving the content from the 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, a thorough analysis of the evaluation system model is provided. The evaluation is based on the *ndnSIM* simulator, an ns-3 module that adopts the *Named Data Networking (NDN)* communication model [1]. A summary of the model can be found in Table 3.

To ease readers relate the results with those presented in other publications, a real network topology, Exodus AS-3967 is used, based on Rocketfuel traces [33]. The topology, shown in Fig.1, consists of 94 nodes, i.e. 39 backbone nodes (blue) and 58 gateway nodes (green). 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 YouTube traffic is determined. However, as the exact simulation of such a scenario is computationally expensive [28], the normalization of some characteristics is necessary so as to adopt the model; both the catalog size $|O|$ and the content store (CS) size cs_i are reduced by a magnitude of 10^4, i.e. from 10^8 [5, 37] to 10^4 and from 10GB [2] to 1MB, respectively. Object sizes follow a normal distribution of mean

Table 3: 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
Producer connectivity degree	$\|D\|$	$\{2,5,11\}$	No. of connections of producer node
Capacity of links	BW	40GB	Available bandwidth
Catalog size	$\|O\|$	$\{1000, 10000\}$	Total no. of objects
Object size	o_i	$\forall o_i, i\varepsilon\|O\| \sim N(10000KB, 9800KB)$	Size of object o_i in KB
Chunk size	Ch	$\{1KB, 10KB\}$	Chunk size in KB
Contents size	$\|C\|$	$\sum_{i=1}^{\|O\|} o_i/Ch$	Total no. of chunks
Cache size	cs_i	$\forall cs_i, i\varepsilon\|N\|, cs_i\varepsilon\{1,10,100,1000\}$	Cache capacity of node i in chunks
Consumers Size	u_i	$\forall u_i, i\varepsilon\|G\| \sim U(100,300)$	No. of users on gateway i
Rank parameter	q	$q\varepsilon\{0.5,5\}$	Rank parameter of the Z-M distribution
Zipf exponent	α	$\alpha\varepsilon\{0.8,1.0,1.5,2.0,2.5\}$	Exponent of the Z-M distribution
Arrival rate	λ	1.0	Exponential request arrival rate
Control window	W	∞	No. of requests able to sent with no reply

10MB and standard deviation 9.8MB [10]. To study the effect of popularity on the performance of the algorithms, a Zipf-Mandelbrot (Z-M) object popularity distribution of $\alpha\varepsilon\{0.8,1.0,1.5,2.0,2.5\}$ and $q\varepsilon\{0.5,5\}$ [5, 36, 11] is defined. Contents in CS are replaced using a Least Recently Used (LRU) policy [14, 28].

Finalizing 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, $Ch = 10KB$ [10, 37]. Request arrivals follow an exponential distribution of $\lambda = 1.0$.

The simulation time of the system model equals to 200 seconds. To avoid out of date results, a convergence time of 100 seconds is set. Hence, evaluation metrics are recorded after the first 100 seconds and every 1 second.

5 Evaluation

Using the system model described in section 4 and the evaluation metrics of cache hit rates, cache eviction rates, content delivery times and hop counts, a report of the evaluation results of the DC, FIX(0.9), ProbCache and Prob-PD algorithms is provided. Each evaluation result corresponds to the mean value of 10 simulation runs under a range of parameters: $\{\alpha, q, cs_i\}$. Due to space limitations and in order to ease readability, only the mean values of the evaluation results are displayed. To be able to compare the caching mechanisms against one another an average value derived from the evaluation results with respect to each parameter is used. Again, due to space limitations, a modification on the names of the algorithms is applied, i.e. *P(0.9), PC* and *PD* for the FIX(0.9), Prob-Cache and Prob-PD algorithms, respectively.

Fig.2 illustrates the performance of the caching mechanisms with regard to the cache hit rates for a range of parameters, i.e. $\alpha\varepsilon\{0.8,1.0,1.5,2.0,2.5\}, q\varepsilon\{0.5,5\}$ and $cs_i\varepsilon\{10,100,1000\}$. According to Fig.2, cache hit rates can be significantly affected by parameter α; the number of cache hits decreases as α increases from 0.8 to 1.0 for $q = 0.5$ and from 0.8 to 1.5 for $q = 5$ and increases again for $\alpha \geq 1.0$ and $\alpha \geq 1.5$, respectively. DC constitutes an exception in the case where $0.8 \leq \alpha \leq 1.0$ and $q = 0.5$, for which the cache hit rates increase.

In a Zipf-Mandelbrot distribution, parameters α and q determine the probability of an object to be requested. As α and q increase, requests get limited to a stricter subset of objects. The behavior of the algorithms suggests that when α lies between the aforementioned values, 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 increase the cache hit rates given the number of consumers u_i on each gateway.

According to Fig.2, PD and PC outperform the rest of the alternatives, with PD performing slightly better than PC; approximately 6.81×10^{-3} cache hit rates on average. DC provides the lowest cache hit rates, i.e. about 0.05 less than PD. An important point that needs to be highlighted is the lack of PD algorithm to provide similar cache hit rates compared to the alternatives as the cache size increases. The reason for this result is that PD algorithm operates regardless of the cache size, i.e. if no more contents are defined as popular, no more contents will be cached. Thus, a higher cache size has no significant impact on the performance of the algorithm after a

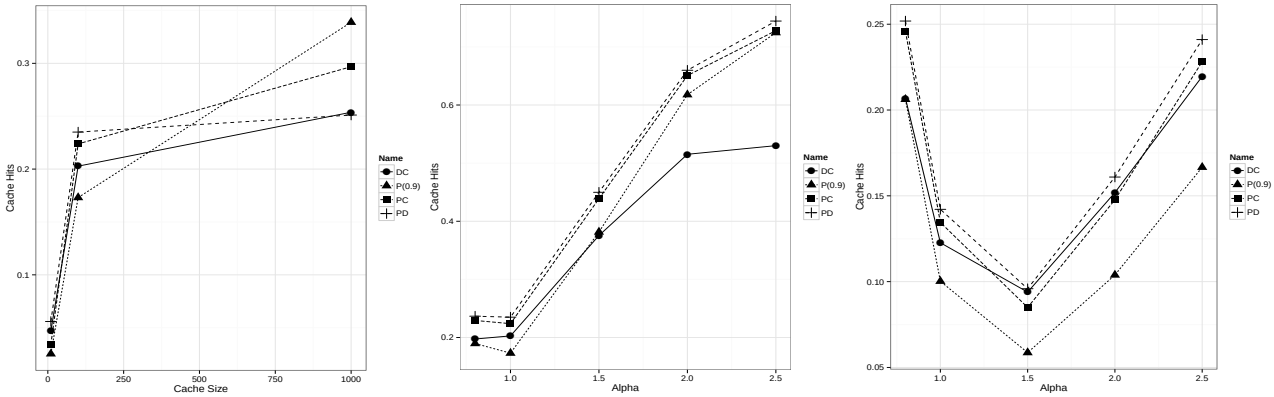

Figure 2: Average cache hit rates for $|D| = 5, Ch = 10KB, |O| = 10.000 : (left.)\, \alpha = 1.0, q = 0.5, cs_i \varepsilon \{10, 100, 1000\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i =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

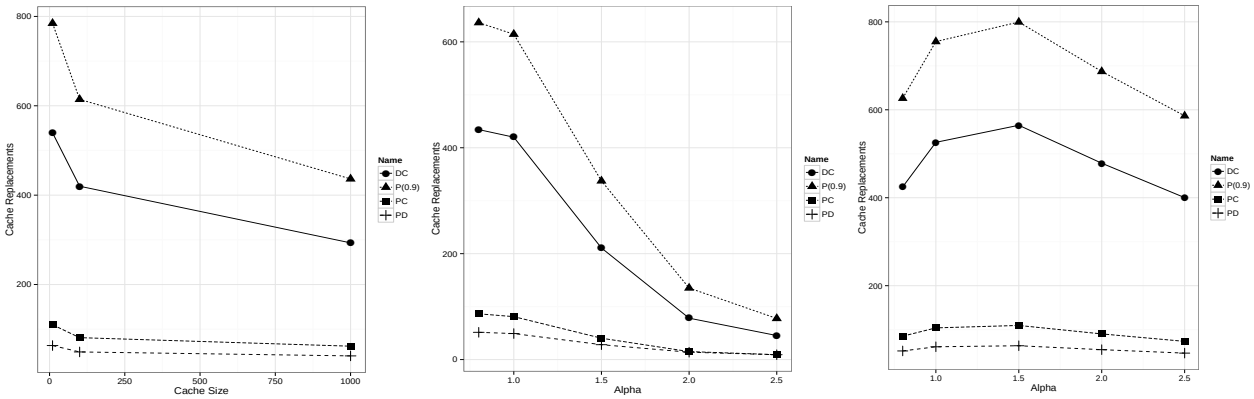

Figure 3: Average eviction rates for $|D| = 5, Ch = 10KB, |O| = 10.000 : (left.)\, \alpha = 1.0, q = 0.5, cs_i \varepsilon \{10, 100, 1000\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$, $(left.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$.

specific threshold. This threshold with regard to the corresponding experiment is estimated at $Ch = 100$ chunks.

If our assumption about PD algorithm is true, cache eviction rates should be relatively low. Fig.3 presents the cache eviction rates of each approach with regard to the same set of parameters. According to the plot, PD corresponds to the lowest cache eviction rates while PC and DC correspond to an approximate increase of 30 and 330 cache evictions of the cache eviction rates recorded for PD. As expected, P(0.9) produces the highest cache eviction rates, i.e. about 500 more than PD.

Finalizing the presentation of the evaluation results for the specific system model we also plot the content delivery times and hop counts for each approach, presented in Fig.4 and Fig.5. As hop count metric is considered to be an estimation of the latency metric, both figures conclude to a similar outcome. Somewhat surprisingly to Fig.2, DC corresponds to the lowest evaluation metric compared to the alternatives. The difference between DC and PC is estimated at 1.9ms for the content delivery times and at 0.085 for the hop counts while the difference between DC and PD is estimated at approximately 3.6ms and 0.17 with regard to the same metrics. The result suggests that caching and network evaluation metrics do not strictly align with each other. As the ultimate goal of caching algorithms is the reduction of latency and network traffic, the outcome suggests that graph-based algorithms may as well correspond to high performance.

Based on the aforementioned figures, i.e. Fig.4 and Fig.5, PD is shown to provide relatively poor performance against the alternatives. The reason for this outcome lies on the calculation of popularity factor P. As each content competes against the sum of requested contents during the time interval Δ_t, a sufficient number of content requests is necessary so as to differentiate a content from the rest contents. Fig.2 indicates that the algorithm may successfully identify a content as popular while Fig.4 indicates that this identification is accomplished on nodes that are distant to the consumers, i.e. closer to the source. Considering that the evaluation scenario is based on a single content producer and that the content requests pattern is relatively scarce, one can conclude that the content request pattern can be clearer as the content requests aggregate towards the source.

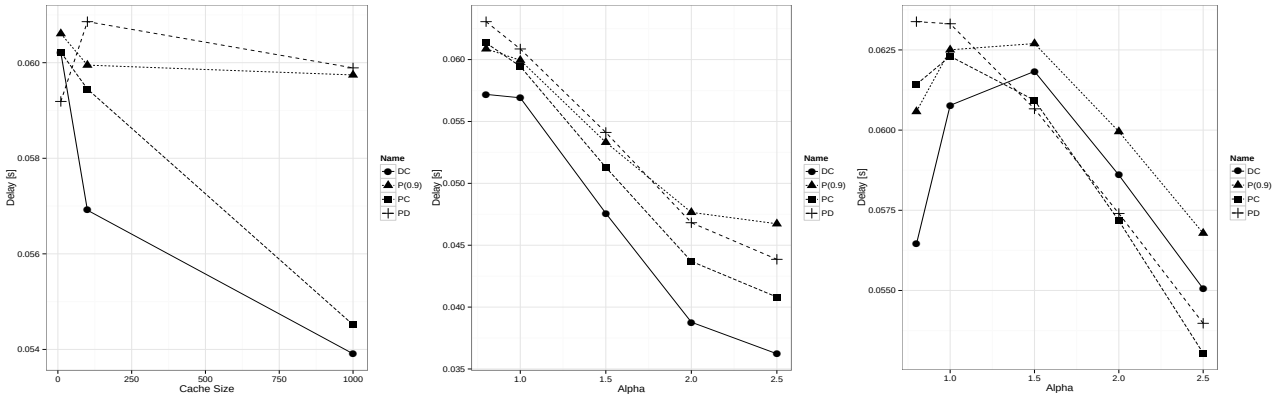

Figure 4: Average delivery times for $|D| = 5, Ch = 10KB, |O| = 10.000 : (left.)\, \alpha = 1.0, q = 0.5, cs_i \varepsilon \{10, 100, 1000\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$.

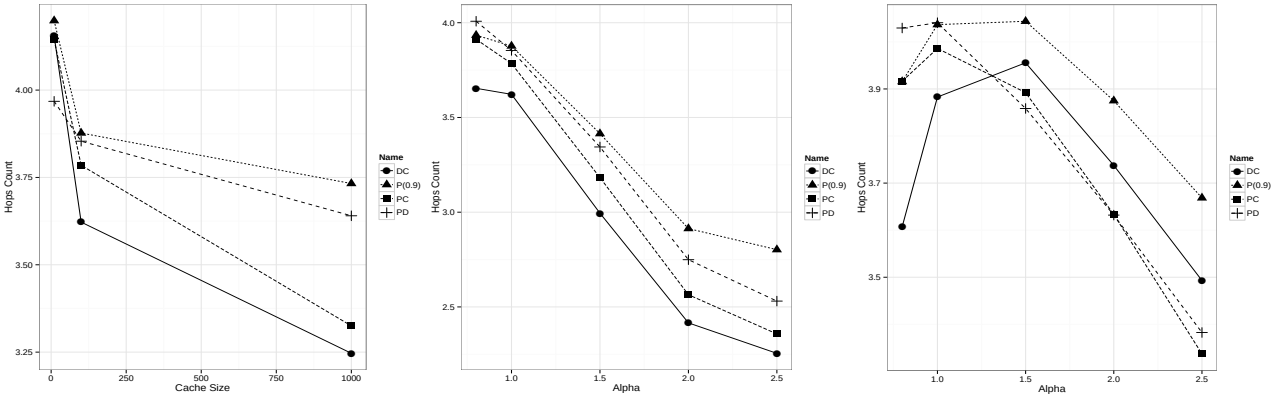

Figure 5: Average hop counts for $|D| = 5, Ch = 10KB, |O| = 10.000 : (left.)\, \alpha = 1.0, q = 0.5, cs_i \varepsilon \{10, 100, 1000\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 100$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 100$.

5.1 Catalog Size Effect

In order to explore whether the performance of PD algorithm is affected by the traffic model, a reduction of the catalog size $|O|$ from 10.000 to 1.000 and a reduction of the cache size cs_i from 100 to 10 are introduced. Similar to the system model described in section 4, both parameters are reduced by the same magnitude.

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, PD corresponds to the highest cache hit rates with a difference of approximately 9.2×10^{-3} and 24.25×10^{-3} against PC and P(0.9), respectively. DC corresponds to a reduction of 33.16×10^{-3} on the cache hit rates recorded for PD. It is worth noting, that in contrast to the previous system model, the cache hit rates do not decrease as the parameter α increases. The main reason for this is a clearer pattern of requests.

Fig.7 plots the cache eviction rates of the algorithms. Comparing the results presented in Fig.7 with these presented in Fig.3, one can observe that the new system model has altered the behavior between DC and P(0.9), with P(0.9) performing the worst on the first system model and DC performing the worst on the second system model. The difference between DC and P(0.9) is estimated to an increase of 3300 cache evictions while the difference between PD and PC is estimated to an increase of 3 cache evictions.

In contrast to the cache eviction rates, the plots for both the content delivery times and hop counts, shown in Fig.8 and Fig.9, are no similar to the ones recorded in the previous evaluation. Fig.8 suggests that the variation of the delivery times of the algorithms is as low as 0.4ms between PD and PC, with the lowest delivery time being 54.5ms, for PD and the highest delivery time being 60ms, for P(0.9). The reported variation is lower than the one recorded in Fig.4. However, the major difference between the figures is the behavior of PD algorithm. Due to the reduction of catalog size $|O|$, content requests get limited to a more narrow range of contents which results in a clearer pattern of requests, even for the nodes close to consumers. The plot of hop counts in Fig.9 follows the same trend as content delivery times in Fig.8, concluding to an approximate average value of 3.71 and 3.77 hop

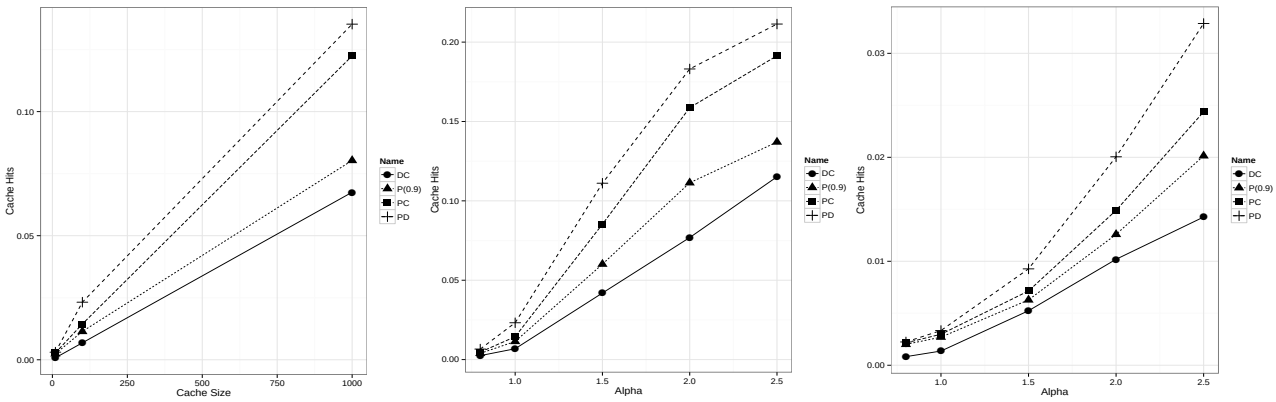

Figure 6: Average cache hit rates for $|D| = 5, Ch = 10KB, |O| = 1.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

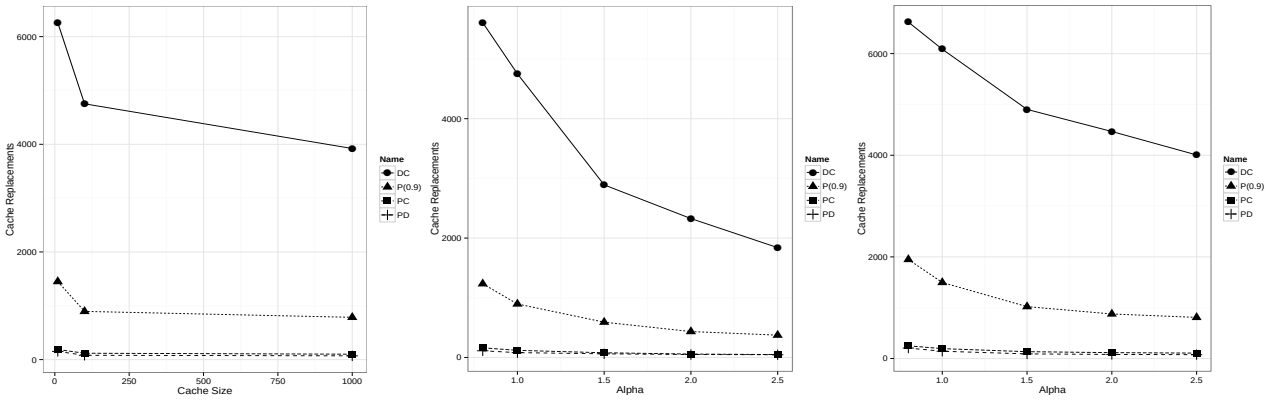

Figure 7: Average eviction rates for $|D| = 5, Ch = 10KB, |O| = 1.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

counts for PD and PC, respectively, and an average value of 3.95 and 4.15 hop counts for DC and P(0.9).

5.2 Producer Connectivity Effect

Recalling the system model presented in section 4, producer installation is based on the metric of connectivity degree $|D|$. Based on the minimum $|D| = 1$ and the maximum $|D| = 14$ recorded in the network, a node with connectivity degree $|D| = 5$ was chosen. To conclude whether the selection of a specific gateway would affect the simulation results, a node with connectivity degree lower and connectivity degree higher than the one used are considered, i.e. $|D| = 2$ and $|D| = 11$, respectively. Due to space limitations and the fact that latency reduction is the ultimate goal of caching algorithms, only the results of content delivery times are displayed, in Fig.8 and Fig.9 for $|D| = 2$ and $|D| = 11$, correspondingly.

According to Fig.3 and Fig.8, the alteration of connectivity degree from $|D| = 5$ to $|D| = 2$ has no significant impact on the behavior of the algorithms; the plots of both graphs, as well as their numerical values, are almost identical, i.e. an increase of about 12ms on the values recorded in Fig.3 is shown. While the behavior of PC, P(0.9) and PD remains similar to the aforementioned figures, Fig.9 suggests that DC exhibits a much different behavior. DC that was shown to perform the best among the alternatives, it now performs the worst. P(0.9) indicates similar performance to DC with their difference being as low as 1.8ms in favor of P(0.9). Finally, PC and PD outperform the rest of the alternatives, with their content delivery times estimated to 51.2ms and 52.7ms.

The behavior of DC lies on the fact that it caches content on the node with the highest connectivity degree. Considering that the maximum connectivity degree in the network is $|D| = 14$ and that the producer installation is performed on a node with $|D| = 11$, results on content delivery times suggest that DC algorithm may have chosen to cache at the node where the producer is installed.

5.3 Chunk Size Effect

Towards analyzing the behavior of the algorithms with regard to network traffic, two additional system models

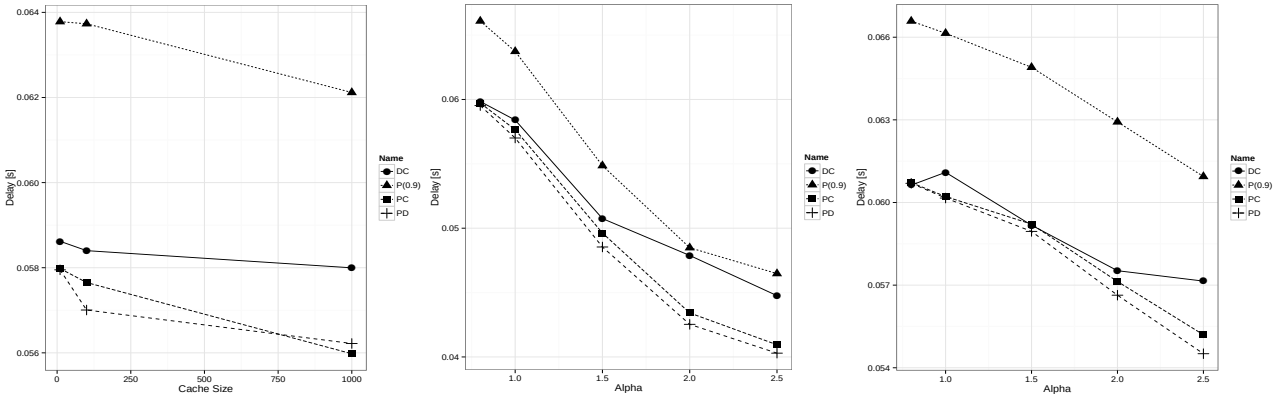

Figure 8: Average delivery times for $|D| = 5, Ch = 10KB, |O| = 1.000$: $(left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

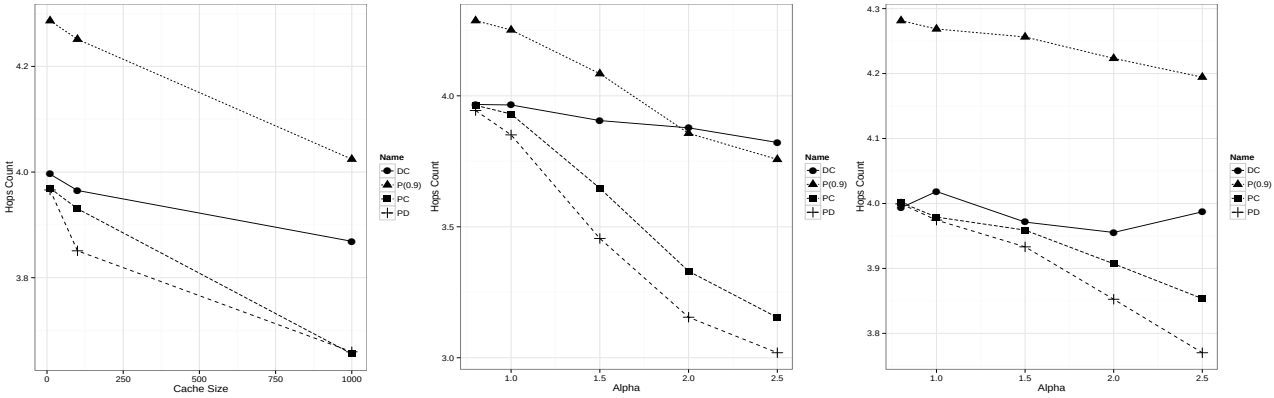

Figure 9: Average hop counts for $|D| = 5, Ch = 10KB, |O| = 1.000$: $(left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

are evaluated by changing the chunk size Ch, from 10KB to 1KB [4]; a system model of $|O| = 10.000$ and $Ch = 1KB$ and a system model of $|O| = 1.000$ and $Ch = 1KB$. Chunk size reduction is as an alternative way to provide a larger number of contents, i.e. $(|O| = 10.000 \times o_i = 10MB/Ch = 1KB) = 10^8$ chunks, in contrast to the previous system model, i.e. $(|O| = 10.000 \times o_i = 10MB/Ch = 10KB) = 10^7$ chunks. The content delivery times of the system models are presented in Fig.12 and Fig.13.

Based on Fig.4 and Fig.12, the behavior of DC and P(0.9) with regard to the different system models is exactly the opposite. DC that was shown to perform the best, it now provides the highest content delivery times, while PD exhibits a comparable performance of approximately 0.9ms difference. Similarly, P(0.9) that was shown to perform the worst among the alternatives, it now performs in rates similar to the current best alternative, i.e. PC with a difference of as low as 0.7ms. Chunk size reduction affects the behavior of PD as well. Based on Fig.4, PD performs close to the worst alternative for $q = 0.5$, while based on Fig.12, PD performs close to the best alternative. On the contrary, based on Fig.4, PD performs close to the best alternative for $q = 5$ while based on Fig.12, PD performs close to the worst alternative. The impact of chunk reduction on the results of PC is not that crucial as the rest of the algorithms.

The examination of the second system model concludes to less significant changes with regard to the behavior of the algorithms. The most noticeable change is the reduction of content delivery times of P(0.9), from 60ms to 54ms while the least noticeable change is the reduction of content delivery times of DC, from 56ms to 55ms. PC, on the other hand, is shown to experience an increase of 1.5ms, which constitutes the worst performance recorded in Fig.13 while PD is shown to outperform the rest of the approaches by 2ms less than the one recorded in Fig.8 and by 1.2ms less than the best alternative, P(0.9). This difference of PD to its best alternative is higher that the one recorded in Fig.8, i.e. 0.4ms.

5.4 Discussion

Based on the results, one can conclude that the performance of on-path caching algorithms can be affected by

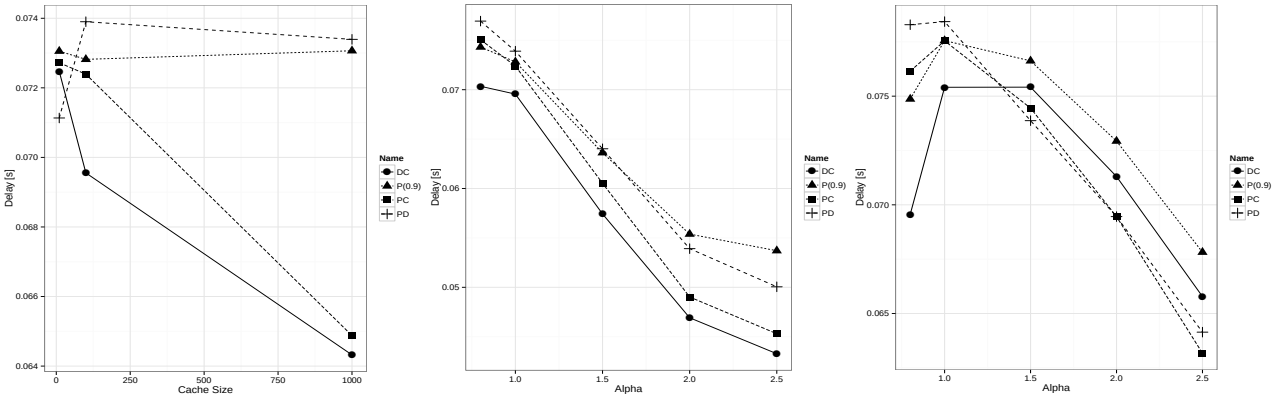

Figure 10: Average delivery times for $|D| = 2, Ch = 10KB, |O| = 10.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

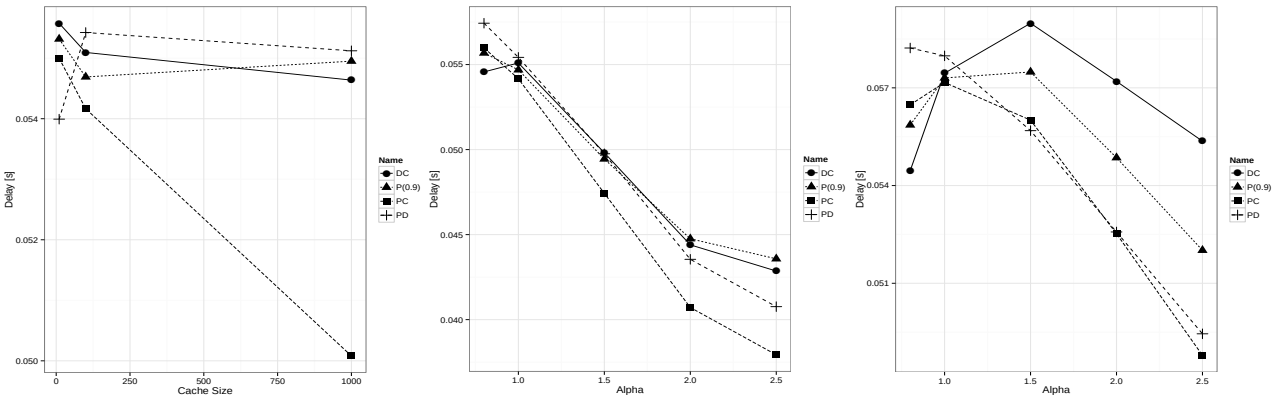

Figure 11: Average delivery times for $|D| = 11, Ch = 10KB, |O| = 10.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

a range of parameters, the most important of which are the catalog size $|O|$ and the chunk size Ch. Both factors are able to significantly affect the traffic model of the system and define a more scarce or a more frequent pattern of requests. The pattern of requests is even more critical to the performance of our algorithm due to the calculation of the popularity factor P, described earlier in this section. Analyzing the impact of parameter $|O|$, we recall the dependence of DC and PC with regard to the metrics of content delivery times and hop counts. More precisely, for $|O| = 10.000$, PC outperforms DC while for $|O| = 1.000$, PC outperforms DC. Similar to catalog size $|O|$, the chunk size Ch concludes to an important dependency of the algorithms with regard to content delivery times, the most critical of which is experienced by DC and P(0.9), i.e. the behavior of the algorithms is opposite to the system model presented in section 4.

The evaluation presented in this paper has helped us identify the suitability of the algorithms under different conditions. According to the results, DC and P(0.9) have been shown to result in a higher dependency on the parameters examined than PC and PD. The outcome suggests that *dependent* algorithms may provide a more robust performance compared to the alternatives. More importantly, the aforementioned evaluation has provided us important information regarding the behavior of PD and its popularity factor P; PD fails to identify a content as popular on nodes where no sufficient information exists, i.e. number of content requests. Based on the catalog size that future Internet architectures are expected to serve, the identification of popularity on a chunk level is considered to be quite challenging. Towards this direction, alternative options for the calculation of popularity are considered, among which a larger, fixed time interval Δt. A larger time interval would probably result in a better categorization of the contents' popularity.

6 Conclusions

In this paper, we have proposed a probabilistic caching algorithm, *Prob-PD*, to enhance performance which we evaluated against the alternatives via simulations. Prob-PD is based on two variables, the content's popularity

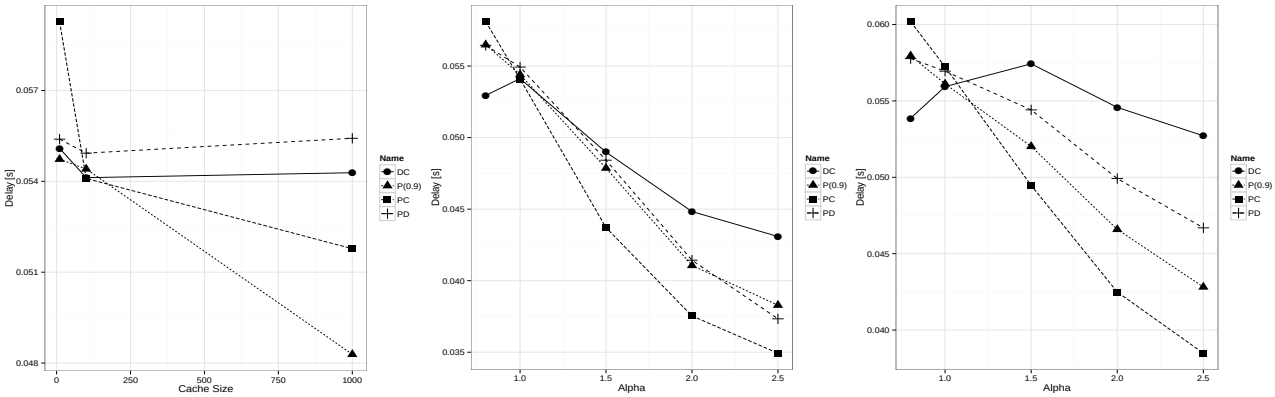

Figure 12: Average delivery times for $|D| = 5, Ch = 1KB, |O| = 10.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

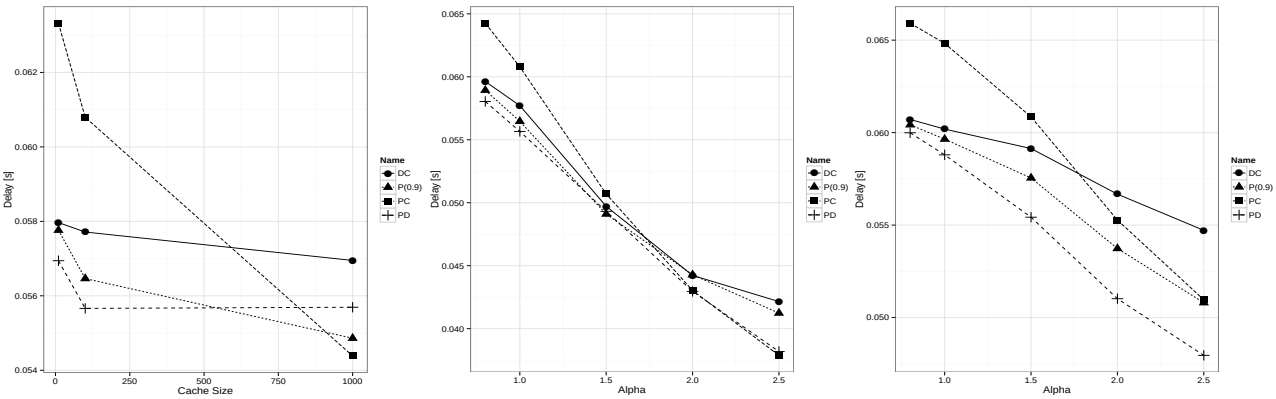

Figure 13: Average delivery times for $|D| = 5, Ch = 1KB, |O| = 1.000 : (left.)\, a = 1.0, q = 0.5, cs_i \varepsilon \{1, 10, 100\}$, $(middle.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 0.5, cs_i = 10$, $(right.)\, \alpha \varepsilon \{0.8, 1.0, 1.5, 2.0, 2.5\}, q = 5, cs_i = 10$.

ratio P, and the distance ratio of each node from the source D. Our results indicate that the performance of an on-path caching algorithm may be considerably affected by the catalog size $|O|$ and the chunk size Ch. More precisely, our approach may provide significant gains if certain conditions are met, such as $|O| \leq 10.000$ or $Ch \leq 10KB$. 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 Acknowledgments

This work has been funded by the Higher Education Authority (HEA) under the Programme for Research in Third-Level Institutions (PRTLI) Cycle 5 and co-funded under the European Regional Development Fund (ERDF).

References

[1] AFANASYEV, A., MOISEENKO, I., AND ZHANG, L. ndnsim: Ndn simulator for ns-3. *Technical Report NDN-0005, Named-Data Networking Project* (October 2012).

[2] ARIANFAR, S., NIKANDER, P., AND OTT, J. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop of the ACM CoNEXT Conference (ReArch '10), Philadelphia, USA, November 2010.*

[3] BALAKRISHNAN, H., LAKSHMINARAYANAN, K., RATNASAMY, S., SHENKER, S., STOICA, I., AND WALFISH, M. 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*, pp. 343–352.

[4] CAROFIGLIO, G., GALLO, M., MUSCARIELLO, L., AND PERINO, D. Modeling data transfer in content-centric networking. In *Proceedings of the 23rd International Teletraffic Congress (ITC'11), San Francisco, CA, USA, September 2011*, pp. 111–118.

[5] CHA, M., KWAK, H., RODRIGUEZ, P., AHN, Y.-Y., AND MOON, S. 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*, pp. 1–14.

[6] CHAI, W., HE, D., PSARAS, I., AND PAVLOU, G. 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*, pp. 27–40.

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

[8] CHO, K., LEE, M., PARK, K., KWON, T., CHOI, Y., AND PACK, S. 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*, pp. 316–321.

[9] DETTI, A., BLEFARI MELAZZI, N., SALSANO, S., AND POMPOSINI, M. Conet: a ontent centric inter-networking architecture. In *Proceedings of the ACM SIGCOMM workshop on Information-Centric Networking (ICN '11), Toronto, Canada, August 2011*, pp. 50–55.

[10] GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC '11), San Diego, CA, USA, October 2007*, pp. 15–28.

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

[12] INDEX, C. Cisco visual networking index: Forecast and methodology 2012-2017. *White paper, CISCO* (May 2013).

[13] IOANNOU, A., AND WEBER, S. Towards on-path caching alternatives in information-centric networks. In *Proceedings of the 39th Conference on Local Computer Networks (LCN '14), Edmonton, Canada, September 2014*.

[14] JACOBSON, V., SMETTERS, D., THORNTON, J., PLASS, M., BRIGGS, N., AND BRAYNARD, R. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09), Rome, Italy, December 2009*, pp. 1–12.

[15] JAMIN, S., JIN, C., KURC, A., RAZ, D., AND SHAVITT, Y. Constrained mirror placement on the internet. In *Proceedings of the 20th IEEE Conference on Computer Communication (INFOCOM '01), Anchorage, AK, USA, April 2001*, pp. 31–40.

[16] JANASZKA, T., BURSZTYNOWSKI, D., AND DZIDA, M. On popularity-based load balancing in content networks. In *Proceedings of the 24th International Teletraffic Congress (ITC'12), Krakow, Poland, September 2012*, pp. 1–8.

[17] KANGASHARJU, J., ROBERTS, J., AND ROSS, K. Object replication strategies in content distribution networks. *Elsevier Journal on Computer Communications 25*, 4 (2002), 376–383.

[18] LAOUTARIS, N., TELELIS, O., ZISSIMOPOULOS, V., AND STAVRAKAKIS, I. Distributed selfish replication. *IEEE Transactions on Parallel and Distributed Systems 17*, 12 (2006), 1401–1413.

[19] LEE, U., RIMAC, I., AND HILT, V. 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*, pp. 179–182.

[20] LI, B., GOLIN, M. J., ITALIANO, G. F., DENG, X., AND SOHRABY, K. 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*, pp. 1282–1290.

[21] LI, J., WU, H., LIU, B., LU, J., WANG, Y., WANG, X., ZHANG, Y., AND DONG, L. 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*, pp. 15–26.

[22] LUA, E., CROWCROFT, J., PIAS, M., SHARMA, R., AND LIM, S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials, 7*, 2 (2005), 72–93.

[23] MAHANTI, A., EAGER, D., AND WILLIAMSON, C. Temporal locality and its impact on web proxy cache performance. *Elsevier Journal on Performance Evaluation 42*, 2 (2000), 187–203.

[24] PAN, J., HOU, Y., AND LI, B. An overview of dns-based server selections in content distribution networks. *Computer Networks 43*, 6 (2003), 695–711.

[25] PASSARELLA, A. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Elsevier Journal on Computer Communications 35* (2012).

[26] PSARAS, I., CHAI, W., AND PAVLOU, G. Probabilistic in-network caching for information-centric networks. In *Proceedings of the 2nd Workshop on Information-Centric Networking, Orlando, FL, USA, March 2012*, pp. 55–60.

[27] RABINOVICH, M., RABINOVICH, I., RAJARAMAN, R., AND AGGARWAL, A. 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*, pp. 101–113.

[28] ROSSI, D., AND ROSSINI, G. Caching performance of content centric networks under multi-path routing (and more). Tech. rep., Telecom ParisTech Ecole, November 2011.

[29] ROSSINI, G., AND ROSSI, D. 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*, pp. 280–285.

[30] SOURLAS, V., FLEGKAS, P., GKATZIKIS, L., AND TASSIULAS, L. Autonomic cache management in information-centric networks. In *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '12), Maui, HI, USA, April 2012*, pp. 121–129.

[31] SOURLAS, V., FLEGKAS, P., PASCHOS, G., KATSAROS, D., AND TASSIULAS, L. Storage planning and replica assignment in content-centric publish/subscribe networks. *Elsevier Journal on Computer Networks 55*, 18 (2011), 4021–4032.

[32] SOURLAS, V., PASCHOS, G., FLEGKAS, P., AND TASSIULAS, L. Caching in content-based publish/subscribe systems. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09), Honolulu, HI, USA, November 2009*, pp. 1–6.

[33] SPRING, N., MAHAJAN, R., AND WETHERALL, D. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking (TON) 32*, 4 (2002), 133–145.

[34] VAKALI, A., AND PALLIS, G. Content delivery networks: Status and trends. *IEEE Journal on Internet Computing 7*, 6 (2003), 68–74.

[35] XYLOMENOS, G., VERVERIDIS, C., SIRIS, V., FOTIOU, N., TSILOPOULOS, C., VASILAKOS, X., KATSAROS, K., AND POLYZOS, G. A survey of information-centric networking research. *IEEE Communications Surveys and Tutorials 16*, 2 (2013), 1–26.

[36] YU, H., ZHENG, D., ZHAO, B. Y., AND ZHENG, W. 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*, vol. 40, pp. 333–344.

[37] ZHOU, J., LI, Y., ADHIKARI, V. K., AND ZHANG, Z.-L. Counting youtube videos via random prefix sampling. In *Proceedings of the 7th ACM Conference on Internet Measurement (IMC '07), Berlin, Germany, November 2011*, pp. 371–380.